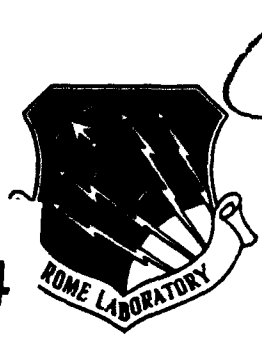
RL-TR-93-239
Final Technical Report
December 1993



CORRELATION FILTER SYNTHESIS USING NEURAL NETWORKS

University of Dayton

Steven C. Gustafson and David L. Flannery

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-93-239 has been reviewed and is approved for publication.

APPROVED:

JOSEPH L. HORNER
Project Engineer

FOR THE COMMANDER

JOHN K. SCHINDLER
Director
Electromagnetics & Reliability Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL (EROP) Hanscom AFB MA 01731. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1993	Final Jun 91 - Apr 93

4. TITLE AND SUBTITLE
CORRELATION FILTER SYNTHESIS USING NEURAL NETWORKS

5. FUNDING NUMBERS
C - F19628-91-K-0014
PE - 61102F
PR - 2305
TA - J7
WU - 40

6. AUTHOR(S)
Steven C. Gustafson and David L. Flannery

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Dayton
Research Institute
300 College Park
Dayton OH 45469-0140

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory (EROP)
80 Scott Rd
Hanscom AFB MA 01731-2909

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-93-239

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Joseph L. Horner/EROP/(617) 377-3841

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Excellent results were obtained using neural networks to synthesize filters for optical correlators, including filters for both cluttered backgrounds and target rotation angles not used in training. The most significant results employed new stretch and hammer neural networks which constitute an important and enduring advance because they train with guaranteed upper bounds on computational effort and generalize with guaranteed lower bounds on smoothness and stability. These results indicate good prospects for training neural networks to synthesize filters for a wide range of target distortions, and this approach has clear advantages compared to searching stored filters.

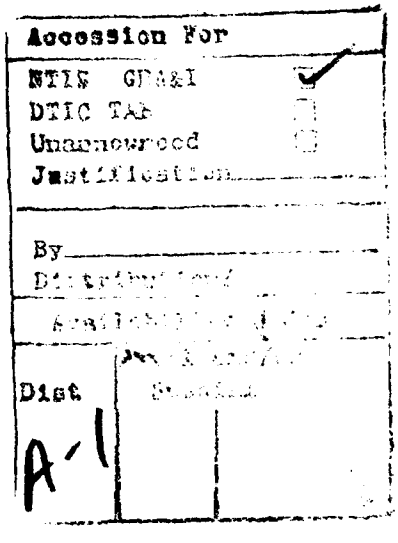
14. SUBJECT TERMS
Neural Networks, Optical Pattern Recognition, Optimizing Algorithms, Target Recognition

15. NUMBER OF PAGES
176

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS



Abstract

Computer simulations were performed that used neural networks to synthesize filters for optical correlators. The synthesized filters were designed to maintain acceptable recognition performance for targets on cluttered backgrounds that were rotated relative to initial (unsynthesized) filters. The most significant results employed new stretch and hammer neural networks which train with guaranteed upper bounds on computational effort and generalize with guaranteed lower bounds on smoothness and stability. These results indicate good prospects for training neural networks to rapidly synthesize filters for a wide range of target distortions. They also indicate possible significant advantages compared to searching stored filters.

1. INTRODUCTION

This section briefly considers the approach and rationale for correlation filter synthesis using neural networks, key results obtained, and papers and presentations produced during the course of the effort.

1.1 Approach and Rationale

The use of optical correlation for the recognition and location of objects (targets) in noisy or cluttered images is well known (see, for example, Refs. 4, 5, and 7 and sources cited in these papers). Briefly, in a typical real-time optical correlator an input image is loaded onto an electrically addressed two-dimensional spatial light modulator (SLM), which is an array (e.g., 128 by 128) of closely spaced active apertures. The output from the input SLM is, for example, an array of binary coherent optical amplitudes that represents the input image. A lens (or system of lenses) forms the Fourier transform of this binary image at a second or filter SLM. The filter SLM consists of an array of binary or ternary states, typically 0° and 180° phase shifts or these two phase shifts plus a zero-amplitude state. The filter SLM states are determined by thresholding the conjugate of the Fourier transform or spatial frequency pattern of a target. For example, if a given spatial frequency complex number lies below the line of slope 45° through the origin in the complex plane it is represented by the 0° state; otherwise it is represented by the 180° state. Another lens or system of lenses forms the Fourier transform of the product of the Fourier transform of the binary image and the array of filter SLM states. If targets having the spatial frequencies represented in the filter SLM are present in the input image, then the final Fourier transform plane has bright spots called correlation peaks at the target locations.

Thus a correction peak indicates the presence of a target and specifies its location. However, if the target is rotated or scaled relative to the filter, or if it undergoes any distortion other than translation, then the correlation peak is degraded in amplitude relative to false peaks due to clutter and noise. To address this issue adaptive correlators that load updated filters into the filter SLM have been designed. For example, if the target rotates relative to the initial filter the correlation peak decreases, and this change is detected by a video camera. New filters corresponding to different target rotations are then loaded onto the filter SLM until the correlation peak is restored. The success of this adaptive feedback approach depends on the availability of a bank of stored filters

corresponding to many target rotations, scales, and other (more difficult) distortions such as those due to aspect angle changes or partial occlusions. It also requires rapid searching of the stored filter bank to find the filter that restores correlation peak degradation; search times of less than 1/30 sec (which correspond to standard video frame rates) may be required.

Correlation filter synthesis is an alternate and possibly more elegant approach to avoiding correlation peak degradation due to target distortions, particularly distortions other than rotation and scale that may be difficult to exhaustively pre-compute. In this approach (see Refs. 4 and 7 and sources cited therein) gray-level pixels from a region of the original input image in the neighborhood of the target location identified by the correlation peak are input to neural network processors (or possibly to fuzzy-logic or genetic algorithm based systems). These processors are trained to produce as output the filter for the current target (as rotated, scaled, or otherwise non-translationally distorted) that yields the best possible correlation peak. There is typically one software-simulated neural network for each filter parameter to be determined, and each network has all target-neighborhood pixel gray levels as its input. Since trained neural networks may be understood as "smart" data interpolators, the stored filter and the filter synthesis approaches have much in common: in the former new filters are found by searching a data bank consisting of the filters themselves; in the latter filters are formed from a distributed data bank that contains neural network interaction strengths or weights.

1.2 Key Results and Outputs

Excellent computer simulation results were obtained using neural networks to synthesize filters for optical correlators when the targets (including targets on cluttered backgrounds) were rotated relative to the filters. The most significant results employed new stretch and hammer neural networks which may constitute an important and enduring advance because they train with guaranteed upperbounds on computational effort and generalize with guaranteed lower bounds on smoothness and stability. These results indicate good prospects for training neural networks to synthesize filters for a wide range of target distortions. They also indicate possible significant advantages compared to searching stored filters.

The technical effort on correlation filter synthesis using neural networks was successful and productive. It supported, wholly or in part, research that produced:

- Four papers submitted to refereed journals (one published in Optics Communications[1], one published in Applied Optics[2], and two under review by IEEE Transactions on Neural Networks[3] and Neural Computation[9]).

- Two conference presentations (SPIE Critical Review, San Jose, November 1991[4] and SPIE OE/Aerospace Sensing, Orlando, April 1992[5]).

- Two Electro-optics Master's Theses[6,7] (one entirely on filter synthesis using neural networks) and an Invention Disclosure[8] on stretch and hammer neural networks.

2. TARGETS, BACKGROUNDS, AND FILTERS

This section discusses the targets, backgrounds, and filters used in computer simulations to investigate correlation filters synthesized by neural networks.

2.1 Truck Targets and Clutter Backgrounds

Figure 1a shows a typical truck target and clutter background used for training neural networks for filter synthesis, and Figure 1b shows a typical binarized truck and background used for correlator input. Ref. 7 presents additional examples of targets and backgrounds and describes the binarization procedure. All targets and backgrounds originated from actual gray-level visible or infrared images.

2.2 BPOF and TPAF Filters

A filter with two phase states (typically 0° and 180°) is a binary phase only filter (BPOF), and a filter with these two states plus the zero-amplitude state is a ternary phase amplitude filter (TPAF). Figure 1c is a BPOF obtained from a binarized Fourier transform of the truck in Figure 1a (which was defined to be at 0° rotation). The binarization was performed using a threshold line angle (TLA, the angle between the positive imaginary axis and a line through the origin) of 0°, which requires that complex numbers in the positive half and negative half of the complex plane were represented by a 0° phase shifts and 180° phase shifts, respectively. Figure 1d is a TPAF obtained from the BPOF by imposing a 10 to 60 pixel radius bandpass (i.e., all filter regions except the annular region between these two radii were opaque) and by defining 9 by 9 binary superpixels such that each superpixel had the same phases as the majority of its interior pixels. It was necessary to define and use filter superpixels to reduce the number of

neural networks required for filter synthesis, thus limiting the computational effort. Ref. 7 presents additional examples of BPOF and TPAF filters.

3. STRETCH AND HAMMER NEURAL NETWORKS

This section discusses the radial basis function neural network that was used to obtain the most significant correlation filter synthesis results.

3.1 Bounds on Computational Effort, Smoothness, and Stability

Stretch and hammer neural networks successfully address two common concerns in using a neural network to solve a practical problem: (1) the time required to train the neural network is often excessive, even for a supercomputer, and (2) the trained neural network often does not generalize effectively enough to solve the problem. For stretch and hammer neural networks guaranteed bounds on computational effort ensure that the maximum numerical precision and number of computational steps required for training can be specified in advance of training. In addition, guaranteed bounds on smoothness and stability, which can also be specified in advance of training, ensure that each neural network output changes by no more than a specified value if the training data are changed by a small amount.

As shown in Figure 2a, stretch and hammer neural networks are feedforward architectures that have separate hidden neuron layers for stretching and hammering in accordance with an easily visualized physical mode. The mean X_j of the training values for each input $x_1, x_2, ..., x_n$ is subtracted from each input at the input neurons. A standard principal components transformation then forms linear combinations of these zero-mean inputs through coefficients (or neural network weights) a_{jk}, where j, k = 1, 2, ..., n. The outputs $u_1, u_2, ..., u_n$ of the stretch neurons are therefore linear transformations of the original inputs that "stretch" these inputs to give them equal "importance". Each hammer neuron f_i has as input all stretch neuron outputs and forms an n-dimensional Gaussian radial basis function of these inputs centered on training point i with standard deviation s_i, where i = 1, 2, ..., m and m is the number of training points. Each hammer neuron output is multiplied by a coefficient c_i to form an output neuron input. The output neuron also has as input a bias b_0 and a linear combination, through coefficients $b_1, b_2, ..., b_n$, of the stretch neuron outputs. Thus the final output y consists of a bias term plus n linear terms proportional to the principal-component-transformed inputs plus m nonlinear terms each proportional to an n-dimensional Gaussian function of these inputs.

3.2 Training Procedures and Testing Results

As discussed in Refs. 2, 3, 5, 7, and 8, training the stretch and hammer neural network consists of (1) transforming the inputs to principal components coordinates, thus determining the weights X_j and a_{jk}, (2) finding an a priori hypersurface such as a least squares hyperplane through the training points, thus determining $b_0, b_1, b_2, ..., b_n$, (3) finding the Gaussian radial basis function standard deviations, thus determining the weights s_i, and (4) finding the Gaussian radial basis function coefficients c_i. The training points are interpolated because the number of basis function coefficients equals the number of training points. The basis function standard deviations are chosen to be as large as possible consistent with maintaining diagonal dominance for the simultaneous linear equations that must be solved to obtain the basis function coefficients. As shown rigorously in Refs. 2, 3, and 5, this choice insures that training example generalization is maximally smooth and stable consistent with unique training in a predeterminable number of steps.

Figure 2b compares stretch and hammer neural network and natural cubic spline results for one-input training examples. The curves are comparable except for sparse training example regions, where the stretch and hammer curve approaches the least squares line. This behavior is desirable: cubic spline curves are the smoothest possible, but they typically exhibit unrealistic deviations from the training examples for extrapolation and prediction.

4. FILTER SYNTHESIS PROCEDURES

This section discusses the most successful procedures for obtaining inputs for neural network correlation filter synthesis and for specifying outputs that reduce the number of separate neural networks required. Many other (less successful) procedures are discussed in Ref. 7.

4.1 Input and Output Specification

The most significant neural network filter synthesis results used 600 separate stretch and hammer neural networks, each with 31 inputs. Each input was the mean gray level in one of 31 radial wedges covering a 9 by 9 pixel region centered on the target, which was located by the correlation peak in a 128 x 128 pixel input scene. The output of each neural network was one of the 600 binary 9 by 9 superpixels in a TPAF, where the

TPAF was superpixelated so that training could be accomplished in a few hours on desktop computers.

4.2 Training Specification

As described in Ref. 7, there were 138 sets of training inputs: 46 with the truck target rotated 0°, 2°, ..., 90° on a 66 (out of 256) gray level background, 46 with these angles on a 142 gray level background, and 46 with these angles on a cluttered background. For each set of training inputs the output for each of the 600 neural networks was one of the binary superpixels in the TPAF for the truck on a blank (0 gray level) background rotated by the training input angle.

5. FILTER SYNTHESIS RESULTS

This section discusses the most significant correlation peak to clutter-ratio results for neural network filters synthesized using the procedures discussed above. Many other results (less significant in terms of their practical potential for correlator systems) are discussed in Ref. 7.

5.1 Correlation Peak to Clutter Ratio Plots

Significant filter synthesis results from Ref. 7 using stretch and hammer neural networks are shown in Figure 3. Here correlation peak to clutter ratio (defined as the highest peak in a 5 by 5 pixel target-centered grid divided by the highest peak in the remainder of the 128 x 128 pixel region) is plotted versus in-plane target rotation angle for three correlation filters: the best possible filter (i.e., the 9 by 9 superpixel TPAF from the Fourier transform of the truck at the input rotation angle, where binarization of the truck for the correlator input is selected for the best peak to clutter ratio), the fixed zero degree filter (i.e., the best possible filter for the truck at a fixed 0° rotation angle), and the filter synthesized by 600 stretch and hammer neural networks.

5.2 Evaluation of Synthesized Filters

Figure 3a shows results for a clutter background not used in training, and Figure 3b shows results for both a clutter background and target rotation angles not used in training (1°, 3°, ..., 89°). Note, that the peak to clutter ratio for the fixed filter falls below 3 dB after less than 3° of target rotation, whereas the neural network synthesized filter

remains above 3 dB in Figure 3a and remains above 3 dB in Figure 3b except at two (of the 89) testing angles.

As discussed in Ref. 7, 27 additional graphs similar to those shown in Figure 3 were produced for different input scene and target rotation angle sampling patterns, different clutter backgrounds, peak to sidelobe instead of peak to clutter ratios, and standard backpropagation (see, for example, R. P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP Mag., Vol. 4, pp. 4-22, 1987) instead of stretch and hammer neural networks. One backpropagation neural network with the same inputs, outputs, and training as the 600 stretch and hammer neural networks used to produce the results shown in Figure 3 yielded better results (i.e., higher peak to clutter ratios for the synthesized filters) than the stretch and hammer neural networks. However, this backpropagation neural network (which had 32 inputs, 600 outputs, and 200 hidden-layer neurons with 50- percent of the interconnections randomly removed) had approximately 17 times more adjustable parameters available per output (although these parameters were not all independent) than the stretch and hammer neural network, and its superior performance may be attributed to this factor. Also, backpropagation neural networks with approximately the same number of adjustable parameters (i.e., weights) as the stretch and hammer neural networks (for which training convergence can be guaranteed) did not converge in training. Finally, the one backpropagation neural network that yielded better results required approximately 40 hours to train on a 486-class 33 MHz desktop computer, whereas the 600 stretch and hammer neural networks required approximately three hours.

6. CONCLUSIONS AND PROSPECTS

The storage of only 102,000 (i.e., 600 times 138 + 31 + 1) parameters for all 600 stretch and hammer neural networks was shown to permit the synthesis of filters that yielded peak to clutter ratios above 3 dB in more than 90 percent of the cases for both clutter backgrounds and target rotation angles not used in training. This generally acceptable performance is particularly significant in view of the fact that neural networks may synthesize suitable filters for target tracking (but not in general for target detection, since an initial correlation peak is required) much faster than "smart" filter search strategies. The synthesis of more than ten million filters per second may be feasible if hardware rather than software simulated neural networks are employed. Stretch and hammer and related basis function neural networks, because of their guaranteed upper bounds on training computational effort and their guaranteed lower bounds on

generalization smoothness and stability, may be ideal for synthesizing filters for a wide range of "difficult" target distortions, including aspect angle and obscuration distortions for which training data may be limited. For these distortions the neural network synthesis approach may have significant advantages compared to searching stored filters.

References

1.* S. C. Gustafson, G. R. Little, and D. M. Simon, "Optical-Resonator-Based Neural Network," *Optics Communications*, Vol. 85, pp. 311-314, 15 Sep 91.

2.* S. C. Gustafson, G. R. Little, J. S. Loomis, and T. S. Puterbaugh, "Optimal Reconstruction of Missing-Pixel Images, *Applied Optics*, Vol. 31, pp. 6829-6830, 10 Nov 1992.

3.* S. C. Gustafson, G. R. Little, and J. S. Loomis, "Generalization with Guaranteed Bounds on Computational Effort, Smoothness, and Stability," submitted to *IEEE Trans. Neural Networks*, 13 Apr 92, invited to revise and resubmit.

4.* D. L. Flannery and S. C. Gustafson, "Adaptive Optical Correlation Using Neural Network Approaches, *SPIE Critical Review* CR-40-02, San Jose, CA, 4 Nov 91.

5.* S. C. Gustafson, G. R. Little, M. A. Manzardo, and T. S. Puterbaugh, "Stretch and Hammer Neural Networks," Proc. SPIE Vol. 1710, pp. 43-52, Orlando, FL, 21 Apr 92.

6. E. G. Olczak, "Neural Networks for the Hybrid Adaptive Correlator," *Electro-Optics M. S. Thesis*, University of Dayton, Dayton, OH, Apr 91.

7.* M. A. Manzardo, "Optical Filter Synthesis Using Artificial Neural Networks," *Electro-optics M. S. Thesis*, University of Dayton, OH, Apr 92.

8. S. C. Gustafson, G. R. Little, J. S. Loomis, T. S. Puterbaugh, and P. G. Raeth, "Stretch and Hammer Neural Network," *Univ. of Dayton Invention Disclosure* No. 116, 8 Jul 91.

9.* S. C. Gustafson, T. A. Rhoadarmer, J. S. Loomis, and G. R. Little, "Comparison of Radial Basis Function and Cardinal Cubic Spline Interpolation," submitted to *Neural Computation*, 31 Mar 93, invited to revise and resubmit.

*A copy of this reference is attached.

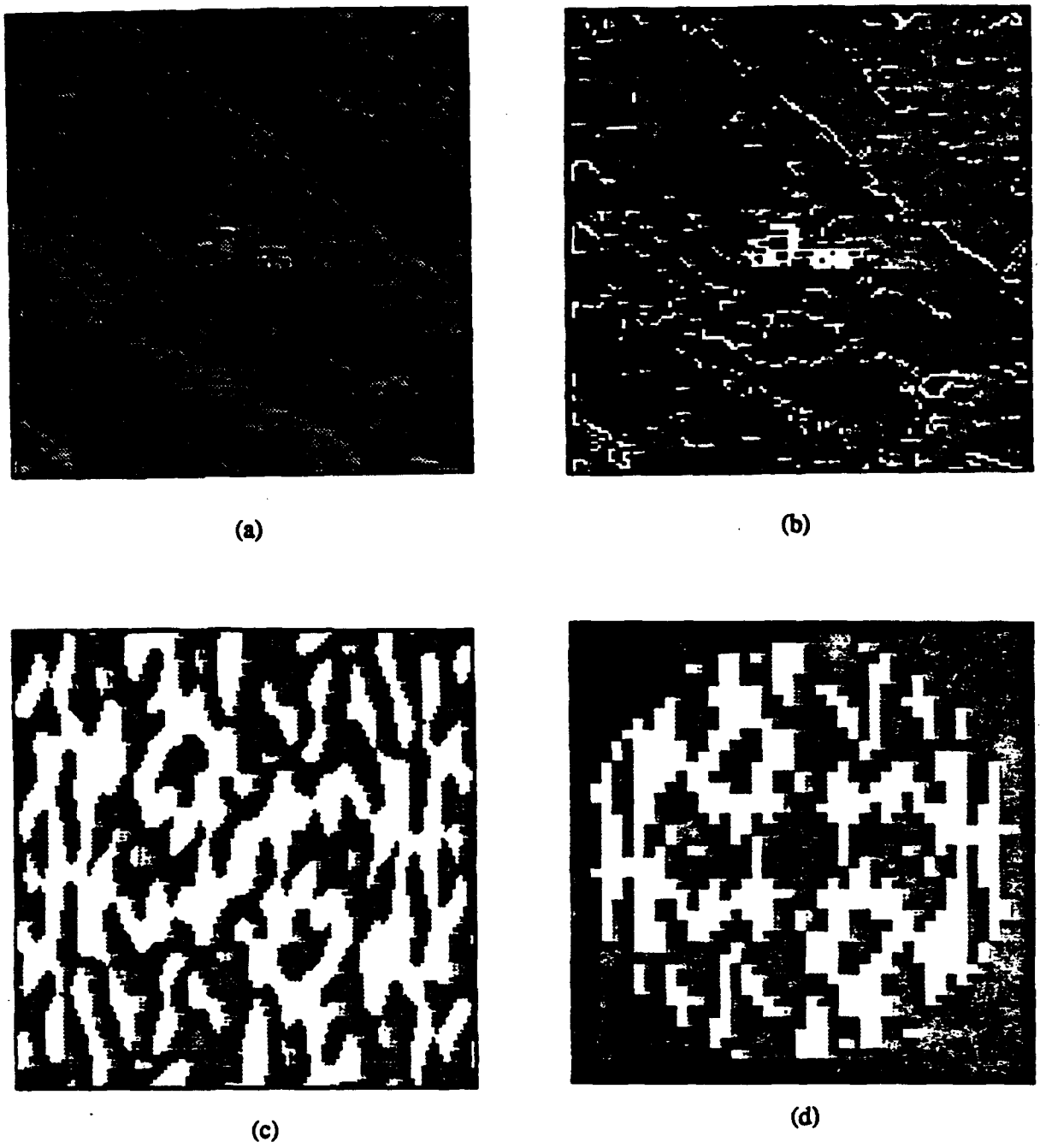
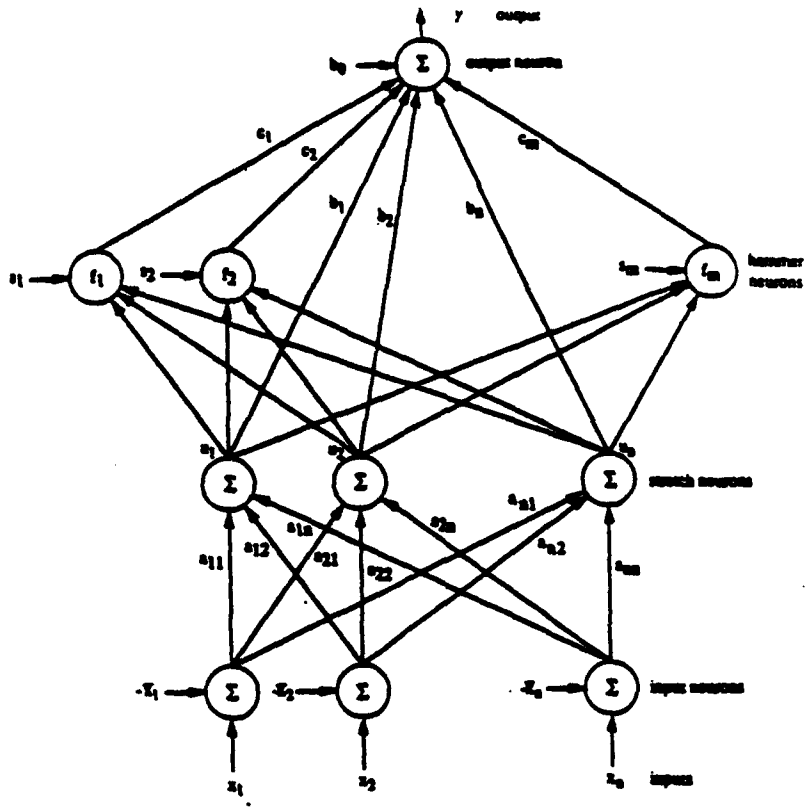


Figure 1. (a) Typical truck target and clutter background used for neural network training. (b) Typical binarized truck and background used for correlator input. (c) BPOF from 0° TLA binarized Fourier transform of truck. (d) TPAF from BPOF using 9 x 9 superpixels and 10-60 pixel radius bandpass.

(a)



(b)

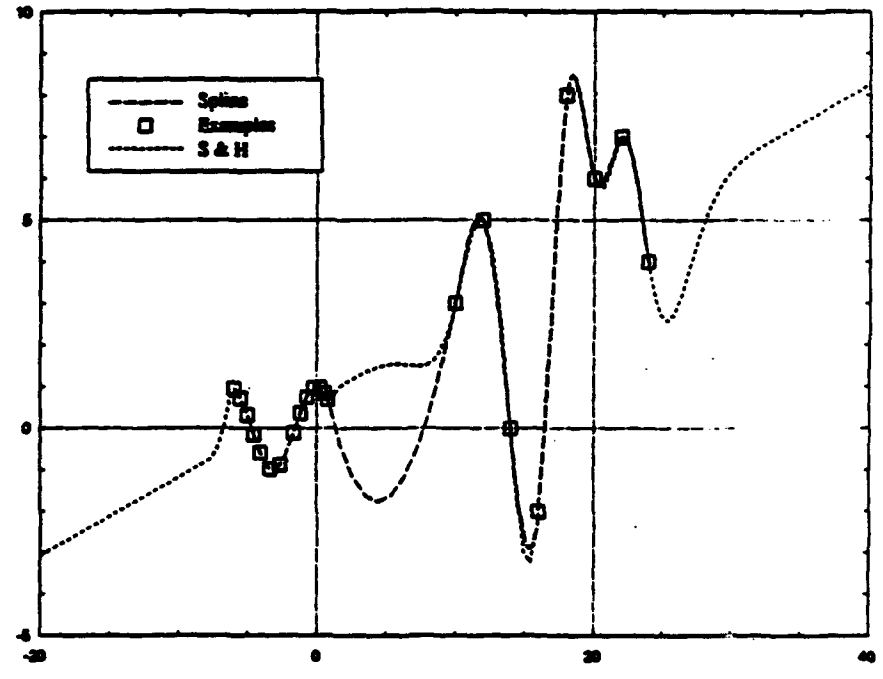
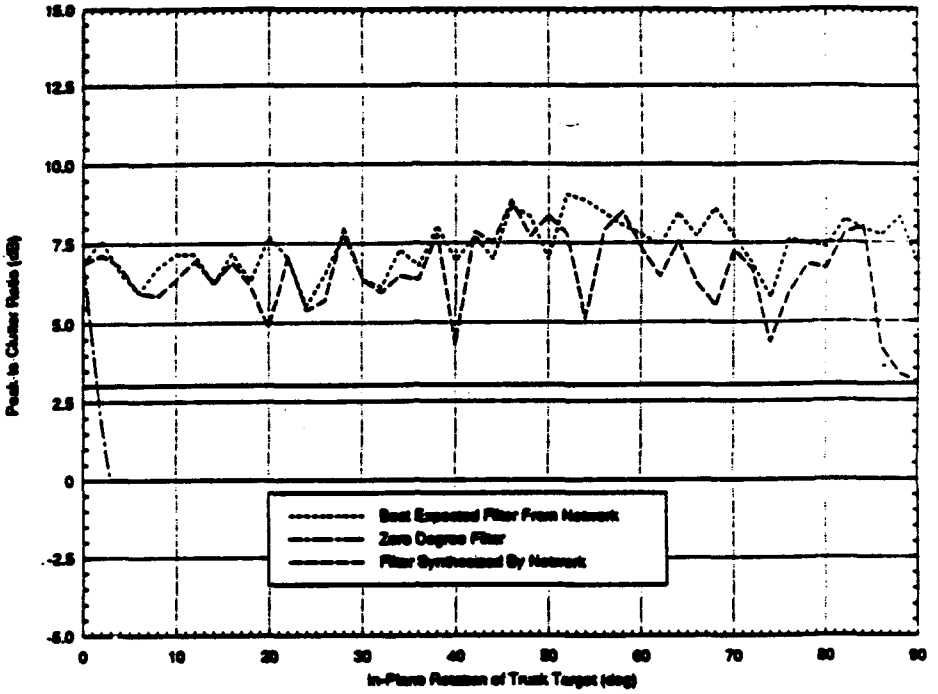


Figure 2. (a) Specification of trained stretch and hammer neural network using Gaussian radial basis functions. (b) Comparison of stretch and hammer neural network and natural cubic spline results for one-input training examples.

(a)



(b)

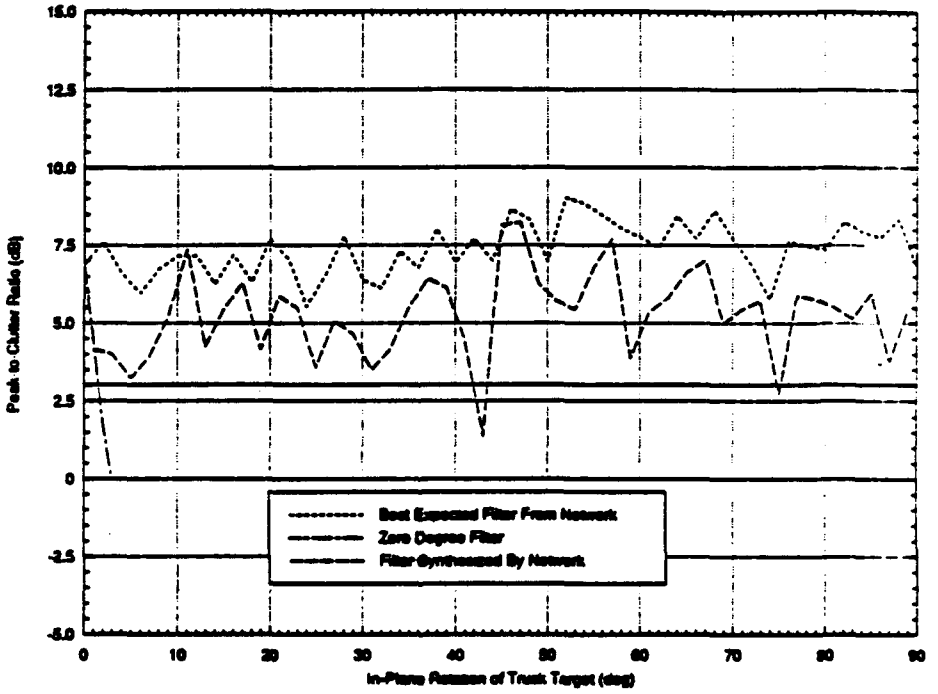


Figure 3. Correlation peak to clutter ratio versus in-plane target rotation angle for three correlation filters: best possible filter, fixed zero degree filter, and filter synthesized by stretch and hammer neural networks, where synthesized filter is for (a) clutter background not used in training and (b) clutter background and rotation angles not used in training.

OPTICS COMMUNICATIONS, Vol. 85, pp. 311-314, 15 Sep 91

Optical-resonator-based neural network

Steven C. Gustafson, Gordon R. Little and Darren M. Simon
Research Institute, University of Dayton, Dayton, OH 45469, USA

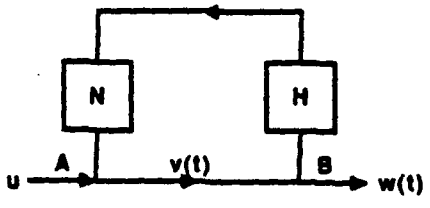
Received 4 June 1991

A neural network model based on an optical resonator is described and its pattern recognition performance is investigated in computer simulations.

This paper describes a neural network model in a form suitable for performing computer simulation experiments and assessing possible optical implementations [1]. The model is consistent with optical resonator designs that may include dynamic holograms and thresholded phase conjugate mirrors [2], and it could be of near-term value in the development of new pattern recognition algorithms.

In many all-optical computing architectures, dynamic holograms are envisioned for interconnection and storage functions and nonlinear components, such as arrays of bistable optical devices or thresholded phase conjugate mirrors, are envisioned for decision operations. The necessary adaptation and feedback interactions between the interconnection and decision components are often achieved by incorporating these components in linear or ring resonators [3–25].

A simple and general formulation of a neural network model consistent with such optical resonator designs may be obtained by well-known methods in which plane wave amplitudes and phases are specified at discrete times separated by the resonator period. In this formulation the model inputs and outputs are complex-element vectors, and a state vector and a hologram matrix evolve in time according to a set of coupled nonlinear difference equations that represent, in general, a high-order threshold logic [26]. The hologram matrix is a func-



$$V(t+1) = N[H(t)v(t)] + Au$$

$$V(0) = Au$$

$$W(t) = Bv(t)$$

$$H(t+1) = \beta H(t) + \gamma M[v(t)v(t)^*]$$

Fig. 1. Neural network model based on an optical resonator. The parameters in the model equations are defined as follows: u, input vector of m complex elements. $v(t)$, internal vector of n complex elements. $w(t)$ output vector of p complex elements. $H(t)$, hologram matrix of $n \times n$ complex elements. $\mathcal{M}[P]$, matrix operator which replaces elements of argument matrix P according to one of two rules: (i) If the waves described by $v(t)$ have evenly spaced propagation directions, all elements of P are replaced by sums along their diagonals. (ii) If the waves described by $v(t)$ have pairwise unequally spaced propagation directions, only the elements of P along the main diagonal are replaced by their diagonal sum. N, nonlinear operator. A, B, complex-element matrices. β, γ, complex constants. $\dagger$, complex transpose. t, discrete time $t = 0, 1, 2, \ldots$

tion of the outer product matrix of the evolving state vector and has a form that depends on the hologram and resonator geometry.

A diagram of the model and equations for the model are given in fig. 1. Note the term in the hologram matrix equation proportional to the outer product matrix of the state vector with either (i) elements on each diagonal or (ii) elements on the main diagonal replaced by their sum. This term may be readily derived for state vector elements as plane waves with either (i) evenly spaced or (ii) pairwise unequally spaced propagation directions, respectively.

For example, consider three plane waves with evenly spaced propagation directions $\theta-\Delta$, θ, and $\theta+\Delta$ that record a diffraction pattern (i.e., a hologram) having amplitude transmittance proportional to the squared magnitude of the sum of the waves. Using $e^{i\theta}$ to represent a plane wave with propagation direction θ, consider the reconstruction of this hologram with waves having the same propagation directions but different complex amplitudes. There are then seven output waves proportional to the terms of

$$[y_1 e^{i(\theta-\Delta)}+y_2 e^{i\theta}+y_3 e^{i(\theta+\Delta)}]\,|x_1 e^{i(\theta-\Delta)}+x_2 e^{i\theta}+x_3 e^{i(\theta+\Delta)}|^2$$

$$=\begin{bmatrix} e^{i(\theta-3\Delta)} \\ e^{i(\theta-2\Delta)} \\ e^{i(\theta-\Delta)} \\ e^{i\theta} \\ e^{i(\theta+\Delta)} \\ e^{i(\theta+2\Delta)} \\ e^{i(\theta+3\Delta)} \end{bmatrix}^T \begin{bmatrix} x_1x_3^* & 0 & 0 \\ x_1x_2^*+x_2x_3^* & x_1x_3^* & 0 \\ x_1x_1^*+x_2x_2^*+x_3x_3^* & x_1x_2^*+x_2x_3^* & x_1x_3^* \\ x_2x_1^*+x_3x_2^* & x_1x_1^*+x_2x_2^*+x_3x_3^* & x_1x_2^*+x_2x_3^* \\ x_3x_1^* & x_2x_1^*+x_3x_2^* & x_1x_1^*+x_2x_2^*+x_3x_3^* \\ 0 & x_3x_1^* & x_2x_1^*+x_3x_2 \\ 0 & 0 & x_3x_1^* \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, \qquad (1)$$

where x_m are the recording wave amplitudes, y_m are the reconstructing wave amplitudes, and T indicates transpose. From eq. (1) the three central output waves are proportional to the terms of

$$p^T.\mathcal{M}[xx^\dagger]y, \quad \text{where } x=\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad y=\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, \quad p=\begin{bmatrix} e^{i(\theta-\Delta)} \\ e^{i\theta} \\ e^{i(\theta+\Delta)} \end{bmatrix}, \qquad (2)$$

where $\dagger$ is the complex transpose operator and $\mathcal{M}$ is an operator that replaces each diagonal with the sum of the elements along that diagonal.

Some comments on the model are: (i) The hologram matrix is self-referenced in that no separate reference beams (e.g., at different angles for different recordings) are involved. (ii) The hologram matrix could at least approximately represent many forms of diffracting structures: thin or thick, amplitude or phase, static or dynamic, reflection or transmission. (iii) The nonlinear operator performs no interconnection operations because it independently replaces each complex element of its argument by another complex element. (iv) The nonlinear operator may incorporate gain or phase conjugation to compensate for wide-angle scattering from the hologram. (v) The nonlinear operator could approximate many types of components, including arrays of bistable optical devices and phase conjugate mirrors with thresholding and gain. (vi) The input and output matrices A and B may represent input and output devices such as beam splitters.

The performance of the model as a pattern recognizer or associative memory for the exclusive-or function was investigated in computer simulations. In this investigation $v(t)$ was a vector of three complex elements, $H(t)$ was a 3×3 matrix of complex elements, $\mathcal{N}$ was an operator that replaced each element of its argument vector by the element squared and divided by the resulting vector magnitude, $\mathcal{M}$ was an operator that replaced each diagonal of its argument matrix with the sum of diagonal elements (as described in the example above for equally spaced propagation directions), and A and B were 3×3 identity matrices.

The model was trained on each of the four exclusive-or function patterns, where the orthogonal complex-plane vectors $(\alpha+i\alpha)/\sqrt{2}$ and $(-\alpha+i\alpha)/\sqrt{2}$ represented 1 and 0, respectively. For training, the initial hol-

ogram matrix $\mathbf{H}(0)$ for each pattern was the 3×3 identity matrix, the first two elements of $s(0)$ were the exclusive-or function inputs, and the third element of $s(0)$ was the exclusive-or function output. The model was allowed to iterate for each pattern until $\mathbf{H}(t)$ no longer changed with t so that four training holograms were generated.

The model was tested with $\mathbf{H}(0)$ set equal to the sum of the four training holograms and with the third element of $s(0)$ set equal to the complex-plane vector i (which is the unit magnitude vector that bisects the angle between the vectors that represent 1 and 0). The model was allowed to iterate for each of the four exclusive-or patterns, and for each case the final third element of s was determined. For appropriately selected values of α, β, and γ, it was found that each of the four final third elements of s had an angle in the complex plane that more closely matched the angle representing a 1 (45°) if this was the correct exclusive-or function output or a 0 (135°) if this was the correct output.

The mean number of iterations required in the testing phase for the correct convergence of the third element of s for the combined four exclusive-or cases was investigated as a function of the parameters α, β, and γ. Convergence was defined to occur when the third element complex-plane angle variation between iterations was less than one part per million. It was found that the mean number of iterations for correct convergence increased as a linear function of the logarithm of γ over at least the range $\gamma=0.002$ to $\gamma=100$. Fig. 2 shows the mean number of iterations for correct convergence versus α and β for $\gamma=0.1$. Note that correct convergence occurs for a wide range of the model parameters. Fig. 3 shows the complex-plane angle of the final minus the initial third element of s versus α for $\beta=0.8$, $\gamma=0.1$, and the four exclusive-or function cases. This angle is the angle of the third element of the vector output of the nonlinear operator $\mathcal{N}$ in fig. 1. In fig. 3, a 1 output is ideally $45°-90°+360°=315°$ and a 0 output is ideally $135°-90°=45°$.

It may be concluded that the optical-resonator-based neural network model can successfully recognize or classify exclusive-or patterns on which it has been trained for a wide range of model parameters. This result is significant because exclusive-or (or inverse exclusive-or) patterns can not be classified using a linear model. Thus a practical pattern classification algorithm based on the optical resonator model may be feasible. Assuming that suitable optical materials and components become available, a long-term consequence could be the development of hardware neural network pattern recognition systems based on optical resonator designs.

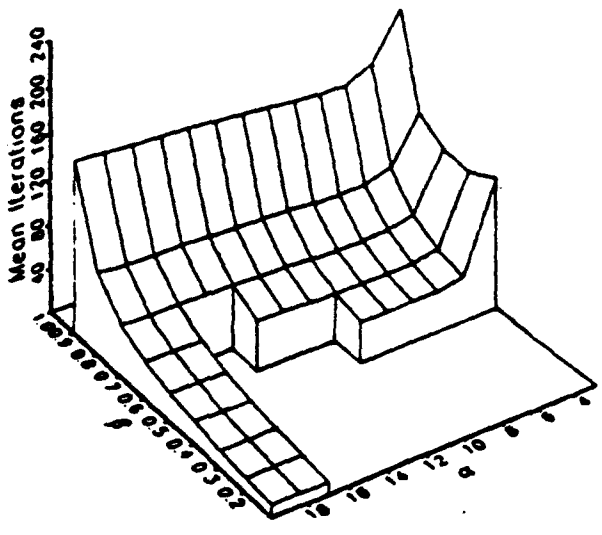


Fig. 2. Mean iterations for correct convergence versus α and β for $\gamma=0.1$. No convergence (correct or incorrect) was obtained in the non-hatched region.

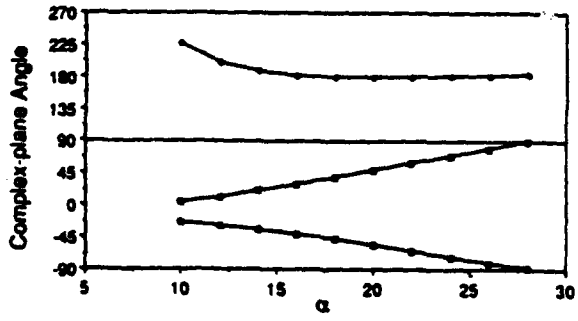


Fig. 3. Complex-plane angle of the final minus the initial third element of s versus α for $\beta=0.8$, $\gamma=0.1$, and the four exclusive-or function cases. The curves identified with open and closed squares represent the cases $(0,0)\rightarrow(0)$ and $(1,1)\rightarrow(1)$ respectively while the single curve identified with solid diamonds represents the two cases $(0,1)\rightarrow(1)$ and $(1,0)\rightarrow(1)$. Acceptable operation, as defined by a 90° complex-plane angle decision boundary, is achieved for $\alpha<25$.

This effort was supported in part by the Center for Artificial Intelligence Applications, a division of the Miami Valley Research Institute, Dayton, Ohio, on US Air Force contract F33615-87-C-1550.

References

[1] S.C. Gustafson and G.R Little, OSA Optical Computing Digest 9 (1989) 300.
[2] G.R. Little and S.C. Gustafson, Univ. Dayton Tech. Rep., UDR-TR-87-113, 1987.
[3] D.Z. Anderson, Optics Lett. 11 (1986) 56.
[4] D.Z. Anderson, Proc. SPIE 613 (1986) 85.
[5] D.Z. Anderson and M.C. Erie, Opt. Eng. 26 (1987) 434.
[6] D.Z. Anderson and M.J. O'Callahan, Proc. SPIE 882 (1988) 181.
[7] M. Cohen, Proc. SPIE 625 (1986) 214.
[8] M. Cohen, Appl. Opt. 25 (1986) 2288.
[9] M.S. Cohen, Proc. SPIE 882 (1988) 122.
[10] S.C. Gustafson and G.R. Little, Proc. SPIE 882 (1988) 83.
[11] D. Psaltis and N. Farhat, Optics Lett. 10 (1985) 98.
[12] K. Wagner and D. Psaltis, Appl. Optics 26 (1987) 5061.
[13] D. Brady, X.G. Gu and D. Psaltis, Proc. SPIE 822 No. 20 (1988).
[14] D. Psaltis, C.H. Park and J. Hong, Neural Networks 1 (1988).
[15] D. Psaltis, D. Brady and K. Wagner, Appl. Optics 27 (1988) 1752.
[16] B.H. Soffer, G.J. Dunning, Y. Owechko and E. Marom, Optics Lett. 11 (1986) 118.
[17] G.J. Dunning, E. Marom, Y. Owechko and B.H. Soffer, Proc. SPIE 625 (1986) 205.
[18] Y. Owechko, G.J. Dunning, E. Marom and B.H. Soffer, Appl. Optics 26 (1987) 1900.
[19] Y. Owechko, Appl. Optics 26 (1987) 5104.
[20] T. Jannson, H.M. Stoll and C. Karaguleff, Proc. SPIE 698 No. 15 (1986).
[21] L.S. Lee, H.M. Stoll and M.C. Tackitt, Optics Lett. 14 (1989) 162.
[22] A. Yariv, S. Kwong and K. Kyuma, Appl. Phys. Lett. 48 (1986) 114.
[23] A. Yariv and S.K. Kwong, Optics Lett. 11 (1986) 1986,
[24] A. Yariv, S.K. Kwong and K. Kyuma, Proc. SPIE 613 (1986) 2.
[25] S.K. Kwong, G.A. Rakuljic, V. Leyva and A. Yariv, Proc. SPIE 613 (1986) 36.
[26] C.L. Giles and T. Maxwell, Appl. Optics 26 (1987) 4972.

APPLIED OPTICS, Vol. 31 , pp. 6829-6830, 10 Nov 1992

Optimal reconstruction of missing-pixel images

Steven C. Gustafson, Gordon R. Little, John S. Loomis, and Todd S. Puterbaugh

The authors are with the Research Institute, University of Dayton, Dayton, Ohio 45469.
Received 27 January 1992.
0003-6935/92/326829-02$05.00/0.

A basis-function technique for reconstructing images with missing pixels is described. This technique yields optimal reconstructed image smoothness in that each basis-function width is maximized consistent with an acceptable level of computational effort.

Key words: *Image reconstruction, image restoration.*

The reconstruction of missing pixels in images is a problem that arises in many contexts. Examples include the uniform-grid resampling of Earth-from-satellite images that have undergone nonlinear geometric transformations to remove motion effects,[1] the restoration of partially obscured nonlinear image boundaries,[2] and the reconstruction of arbitrary-view images from selected-view data in tomography.[3,4] Typical approaches to the reconstruction problem include the use of interpolation or approximation techniques, such as bilinear interpolation or cubic B splines. However, these techniques usually require that all pixels be located on a uniform grid, and they typically yield reconstructed pixel values that are not consistent with known image-formation processes, such as processes modeled by the convolution of Gaussian functions with impulse functions at the pixel locations.

Radial basis-function interpolation and approximation techniques avoid these limitations,[5] but they typically introduce two major concerns: (1) the specification of the extent or the width of the basis functions after their form has been selected consistent with known image-formation processes and (2) the limitation of the level of computational effort required (in both precision and number of computational steps) to obtain the basis-function coefficients, particularly if the number of known pixels is large. As shown below, basis-function techniques can be designed to address these concerns: after the form of the basis functions has been selected consistent with a priori knowledge, optimal reconstructed-image smoothness is achieved in that each basis-function width is maximized consistent with an acceptable level of computational effort. Maximizing basis-function widths may be related to the optimal selection of smoothing parameters in image restoration by regularization.[6]

A typical reconstruction task and an optimal (in the sense indicated above) interpolation approach are as follows. An image has known pixel values z_i (gray level or binary) at locations (x_i, y_i), with $i = 1, 2, \ldots, n$, and unknown (i.e., missing) pixel values z_k at locations (x_k, y_k), with $k = n + 1, n + 2, \ldots, n + m$. Ideally, small clusters of unknown pixels are surrounded by large clusters of known noise-free pixels so that interpolation is appropriate. The basis functions $f_i(x, y) = f_i(x - x_i, y - y_i, v_i)$ are used to fit

$$z(x, y) = c_1 f_1(x, y) + c_2 f_2(x, y) + \cdots + c_n f_n(x, y) \quad (1)$$

to the data $z_i(x_i, y_i)$, where for given (x, y) the magnitude $|f_i(x, y)|$ increases monotonically as the positive width parameter v_i increases and as $|f_i(x, y)|$ approaches zero as either $|x - x_i|$ or $|y - y_i|$ becomes large. If we desire, v_i may be the second moment of $|f_i(x, y)|$, v_i may be a constant independent of i, and $f_i(x, y)$ may equal $f_i(r_i, v_i)$ with $r_i = [(x - y_i)^2 + (y - y_i)^2]^{1/2}$ so that the basis functions are radial. With the functional form of $f_i(x, y)$ specified, the v_i are obtained by solving the n independent nonlinear equation

$$|f_{ii}| - \sum_{j=1}^{n} |f_{ij}| = \epsilon, \qquad i = 1, 2, \ldots, n, \quad (2)$$

where $f_{ij} = f_i(x_j, y_j)$, $j = 1, 2, \ldots, n$, and ϵ is a positive constant that specifies the degree of diagonal dominance of the matrix $F = |f_{ij}|$; thus ϵ, as discussed below, limits the level of computational effort. The basis-function coefficients c_i are then determined by solving the n simultaneous linear equations in n unknowns:

$$z_j = c_1 f_{1j} + c_2 f_{2j} + \cdots + c_n f_{nj}, \qquad j = 1, 2, \ldots, n. \quad (3)$$

Finally, the unknown pixel values z_k are determined by substituting the pixel locations (x_k, y_k) for (x, y) in Eq. (1).

Since the number of pixels may be large, an assessment of the computational effort involved in solving Eqs. (2) and (3) is required. First, note that Eq. (2) specifies n independent nonlinear equations in one positive variable, and thus each of these equations may be solved separately by using standard methods. Second, note that Eq. (3) specifies a linear system, and thus the computational effort required to obtain a solution depends on a condition number of the matrix F. The two-norm condition number κ_2, which equals the square root of the ratio of the largest to the smallest eigenvalue of the product of F^T and F, typically controls the required numerical precision and the number of computational steps independent of the algorithm (iterative or direct with iterative improvement) used to obtain a solution.[7] This condition number may be limited to an acceptably small value by specifying a sufficiently large

value for ϵ in Eq. (2). A result obtained by Varah[8] who used standard matrix norm notation, is $\| F^{-1} \|_\infty \leq 1/\epsilon$. Combining this result with the expressions[7] $\kappa_2 = \| F \|_2 \| F^{-1} \|_2$, $\| F \|_\infty = \max_i \sum_j |f_{ij}|$, and $\| B \|_2 \leq \sqrt{n} \| B \|_\infty$ for any $n \times n$ matrix B, and noting from Eq. (2) that the second expression equals $2\phi - \epsilon$, where $\phi = \max |f_{ik}|$, we may conclude that $\kappa_2 \leq n(2\phi - \epsilon)/\epsilon$. Thus ϵ may be used to limit κ_2, which, as indicated above, typically controls the numerical precision and the number of computational steps required to solve for the coefficients $c_1, c_2, \ldots, c_n$. Significantly, the degree of diagonal dominance of F may be adjusted to yield optimal interpolation surface or reconstructed-image smoothness in that each basis-function width parameter v_i is maximized consistent with an acceptable level of computational effort.

In the same sense that thin-plate spline interpolation has a bending model in which an elastic plane is deformed into contact with the data points, the basis-function interpolation technique described above has a hammering model in which a malleable plane is similarly deformed. In the hammering model the locations of the known and unknown pixels are marked on a malleable plane surface. Using a large number of small strikes to smoothly deform the surface, we direct the hammering at the known pixel locations. For hammering at each known pixel location the number of strikes per unit area, which is proportional to $f_i(x, y)$, depends on the distance from the known pixel location. The dependence is such that the number of strikes per unit area at each known pixel location is less (by an amount proportional to ϵ) than the sum of the number of strikes per unit area at all other known pixel locations (thus ensuring the diagonal dominance of F). Subject to this dependence, the number of hammering strikes is adjusted so that each known pixel location on the malleable plane surface is deformed normal to the plane by an amount proportional to its gray level.

Several comments on the hammering restoration technique follow. First, as indicated above, this technique is applicable when the known pixels are not located on a regular grid, in which case conventional techniques such as cubic splines typically cannot be used. Second, although Gaussian radial basis functions $f_i(r_i, v_i) = \exp(-r_i^2/2v_i)$ are consistent with many image formation processes, wavelet basis functions[9] may also be appropriate. Third, although well-known techniques such as two-dimensional polynomial interpolation have been shown to exhibit singularities for nonuniform sampling,[10] it has been proved that for a wide class of radial basis functions the matrix F is nonsingular without the constraints of Eq. (2) and regardless of the value of v_k.[11] However, nonsingularity does not ensure acceptably small matrix condition numbers: even for provably nonsingular matrices the level of computational effort required to obtain basis-function coefficients is typically unfeasible for sufficiently large matrices.[12,13] Fourth, the hammering interpolation technique has a neural-network interpretation in which the inputs x and y are connected to neurons that implement the basis functions, and the neuron outputs pass through weighted connections to an accumulator that forms the output z. Training the neural network involves finding the connection weights and the basis-function width parameters, and finding these parameters requires multivariable nonlinear optimization for typical basis-function neural networks. However, the technique described above requires only univariate nonlinear optimization to obtain the maximum basis-function width parameters consistent with an acceptable level of computational effort.

This work was supported in part by the U. S. Air Force Rome Laboratory, the Miami Valley Research Institute, and the Martin Marietta Corporation.

References

1. R. Bernstein, "Digital image processing for remote sensing," IBM J. Res. Devel. **20**, 40–57 (1976).
2. D. Pagliaroni and A. K. Jain, "A control point theory for boundary representation and matching," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing* (Institute of Electrical and Electronics Engineers, New York, 1985), 1851–1854.
3. M. Ravichandran and F. C. Gouldin, "Reconstruction of smooth distributions from a limited number of projections," Appl. Opt. **27**, 4084–4097 (1988).
4. S. S. Cha and H. Sun, "Tomography for reconstructing continuous fields from ill-posed multidirectional interferometric data," Appl. Opt. **29**, 251–258 (1990).
5. T. Poggio nd F. Girosi, "Networks for approximation and learning," Proc. IEEE **78**, 1481–1497 (1990).
6. A. M. Thompson, J. C. Brown, J. W. Kay, and D. M. Titterington, "A study of methods of choosing the smoothing parameter in image restoration by regularization," IEEE Trans. Pattern Anal. Mach. Intell. **13**, 326–339 (1991).
7. G. H. Golub and C. F. VanLoan, *Matrix Computations*, 2nd ed. (Johns Hopkins U. Press, Baltimore, Md., 1989).
8. J. M. Varah, "A lower bound for the smallest singular value of a matrix," Linear Algebra Its Appl. **11**(3), 3–5 (1975).
9. S. G. Mallat, "Multifrequency channel decomposition of images and wavelet models," IEEE Trans. Acoust. Speech Signal Process. **37**, 2091–2110 (1989).
10. A. Zakhor and G. Alvstad, "Two-dimensional polynomial interpolation from nonuniform samples," IEEE Trans. Acoust. Speech Signal Process. **40**, 169–180 (1992).
11. C. A. Micchelli, "Interpolation of scattered data: distance matrices and conditionally positive definite functions," Construct. Approx. **2**, 11–22 (1986).
12. F. Girosi and T. Poggio, "Networks for learning," in *Neural Networks*, P. Antognetti and V. Milntinovic, eds. (Prentice-Hall, New York, 1991), pp. 110–154.
13. N. Dyn, "Interpolation of scattered data by radial functions," in *Topics in Multivariate Approximation*, C. K. Chui and L. L. Schumaker, eds. (Academic, New York, 1987), pp. 47–61.

GENERALIZATION WITH BOUNDS ON COMPUTATIONAL EFFORT, SMOOTHNESS, AND STABILITY

Steven C. Gustafson, Gordon R. Little, and John S. Loomis,
Research Institute, University of Dayton
Dayton, OH 45469-0140

Abstract---Generalization is considered in the context of data interpolation, extrapolation, and approximation. For interpolation using certain functional forms it is shown that upper bounds may be placed on required numerical precision and number of computational steps, and it is also shown that lower bounds may be placed on certain measures of interpolation smoothness and stability. These results are obtained for radial or other basis function interpolation using a diagonal dominance criterion for the matrix whose inverse determines the basis function coefficients. The diagonal dominance criterion is particularly appropriate for applications where extrapolated values must asymptotically approach an a priori function, and this criterion also provides a justifiable solution to the problem of selecting basis function parameters.

Generalization realized as data interpolation, extrapolation, or approximation may be performed with upper bounds on computational effort and lower bounds on smoothness and stability. Such generalization can be carried out for certain definitions of the bounded quantities and, in particular, for interpolation using certain functional forms, including forms that asymptotically approach an a priori function, as required for applications such as image reconstruction.

As an example, consider data that consists of m input-output points (x_i, y_i), where the inputs x_i are length n vectors, the outputs y_i are scalars, and i, j = 1, 2, ..., m. These data points are to be interpolated and extrapolated using the function $f(x) = \sum_j c_j \exp[-(x - x_j)^2/\sigma_j^2]$ obtained by convolving $c_j\delta(x - x_j)$, which is an impulse function centered on the jth data input vector, with $\exp(-x^2/2\sigma_j^2)$, which is a Gaussian radial basis function with standard deviation σ_j, and summing the results for all j. For interpolation the coefficients c_j must be such that $y_i = f(x_i)$, and thus they are obtained by solving m simultaneous linear equations in m unknowns $y_i = \sum_j c_j a_{ij}$, where $a_{ij} = \exp[-(x_i - x_j)^2/2\sigma_j^2]$.

The parameters σ_j may be selected such that the matrix $A = \{a_{ij}\}$ is diagonally dominant by a positive amount ε for each column, i.e., so that $\varepsilon = a_{ii} - \Sigma_{i \neq j} a_{ij}$ for all j. Using standard matrix norm notation, it is well known that the ∞-norm of A is $\|A\|_\infty = \max_i \Sigma_j |a_{ij}|$ and that for any n x n matrix B the ∞-norm and the 2-norm are related by $\|B\|_2 \leq \sqrt{n}\ \|B\|_\infty$. Also, it has been shown that the ∞-norm of A^{-1} is $\|A^{-1}\|_\infty \leq 1/\varepsilon$ [Varah, 1975]. Using $a_{ii} = 1$, the above expressions indicate that the 2-norm condition number of A is $\kappa_2 = \|A\|_2 \|A^{-1}\|_2 \leq n(2 - \varepsilon)/\varepsilon$.

The computational effort defined by the numerical precision and the number of iterative-improvement computational steps required to obtain the c_j has upper bounds that increase with κ_2. Suppose that κ_2' is the maximum acceptable 2-norm condition number for A. Then smoothness defined by $(\Sigma_j \sigma_j^2/m)^{1/2}$ is maximized by selecting the σ_j such that $n(2 - \varepsilon)/\varepsilon = \kappa_2'$. This selection requires the solution of m independent nonlinear equations (one for each unknown σ_j), but the iterative-improvement solution of the m simultaneous linear equations $y_i = \Sigma_j c_j a_{ij}$ asymtotically dominates the required number of computational steps as m becomes large. Finally, stability defined by $1/r_c$ is bounded by $1/r_c \geq [(\kappa_2' r_y)^{-1} - 1]/2$ for $\kappa_2' r_y < 1$, where $r_c = [\Sigma_i (c_i' - c_i)^2 / \Sigma_i c_i^2]^{1/2}$ is the fractional root-mean-square coefficient change, $r_y = [\Sigma_i (y_i' - y_i)^2 / \Sigma_i y_i^2]^{1/2}$ is the fractional root-mean-square data output change, and the c_i change to c_i' if the y_i change to y_i' [Golub and Van Loan, 1989].

For this example, generalization is thus achieved with least upper bounds on computational effort and greatest lower bounds on smoothness and stability if the σ_j are selected such that A has diagonal dominance $\varepsilon = 2n/(n + \kappa_2')$. This example may be modified to address approximation, multiple outputs, norms other than the 2-norm, other definitions of computational effort, smoothness, and stability, and non-Gaussian or non-radial basis functions provided that such functions decrease as any of their independent variables increase.

In the same sense that thin plate spline interpolation has a "bending" model in which an elastic plane is deformed into contact with the data points, the above example has a "hammering" model in which a malleable plane is similarly deformed. In the hammering model numerous small strikes are directed at each x_i with Gaussian precision $\exp[-(x - x_i)^2/2\sigma_j^2]$ such that the plane is smoothly deformed into contact with the data points. The hammering standard deviations σ_j are selected such that the ratio of the strike density at x_i to the sum of strike densities at $x_{j \neq i}$ (i.e., the ratio of hits to misses) is

at least $1/\varepsilon = (n + \kappa'_2)/(2n)$. Here κ_2' bounds (1) the numerical precision and number of iterative-improvement computational steps required to determine the relative number of strikes at each x_i, (2) the stability with respect to changes in y_i of this determination as measured by $1/r_c$, and (3) the smoothness of the hammered surface as measured by the root-mean-square of the σ_j.

The diagonal dominance criterion permits the acceptance of larger condition numbers for A than criteria [Narcowich and Ward, 1991] that apply to other classes of basis functions, e.g., radial basis functions that increase with radius. By some criteria such radial functions interpolate more smoothly than basis functions that decrease with radius, but the latter functions are appropriate for problems in which extrapolated values must approach an a priori function with distance from the data inputs, as may be required for image reconstruction applications [Gustafson et al., 1992]. Also, in applications for which the process that produced the data is unknown, generalization that smoothes the data by convolution with a Gaussian or similar decreasing-with-radius function may be justified. Finally, the diagonal dominance criterion provides a justifiable solution to the problem of basis function parameter selection [Poggio and Girosi, 1990], i.e., width parameter such as σ_j are maximized such that the largest acceptable condition number is not exceeded.

1. G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins Univ. Press, 1989.

2. S. C. Gustafson, G. R. Little, J. S. Loomis, and T. S. Puterbaugh, "Optimal Reconstruction of Missing Pixel Images," submitted to *Applied Optics*, Jan. 1992.

3. F. J. Narcowich and J. D. Ward, "Norms of Inverses and Condition Numbers for Matrices Associated with Scattered Data," *J. Approximation Theory*, Vol. 64, pp. 69-94, Jan. 1991.

4. T. Poggio and F. Girosi, "Networks for Approximation and Learning," *Proc. IEEE*, Vol. 78, pp. 1481-1497, Sep. 1990.

5. J. M. Varah, "A Lower Bound for the Smallest Singular Value of a Matrix," *Linear Algebra and Its Applications*, Vol. 11, pp. 3-5, Jan. 1975.

SPIE Critical Review CR-40-02, San Jose, CA, 4 Nov 91

ADAPTIVE OPTICAL CORRELATION USING NEURAL NETWORK APPROACHES

David L. Flannery and Steven C. Gustafson
Research Institute, University of Dayton
Dayton, Ohio 45469-0140

Abstract

This paper reviews work on binary phase-only (BPOF) and ternary phase-amplitude (TPAF) correlation and highlights recent investigations of neural network approaches for augmenting correlation-based hybrid (optical/electronic) automatic target recognition systems. The theory and implementation of BPOF and TPAF correlation using available spatial light modulators is reviewed, including recent advances in smart TPAF formulations. Results showing the promise of neural networks for enhancing correlation system operation in the areas of estimating distortion parameters, adapting filters, and improving discrimination are presented and discussed.

1. INTRODUCTION

Coherent optical correlation and artificial neural network approaches have independently shown promise for pattern recognition, including automatic target recognition (ATR), which is an important military application area. This paper reviews the concepts and progress of investigations that combine these two approaches and that are motivated by the potential for retaining the strengths while bypassing the weaknesses of each approach.

A leading candidate architecture for ATR using optical correlation is the hybrid adaptive correlator (HAC) concept depicted in Figure 1. The HAC consists of a rapid sequential (i.e., "real-time") correlation module imbedded in an overall electronic system that controls the cycle of operation, including the selection of appropriate correlation filters from a large bank of pre-computed "smart" (i.e., combining both distortion-invariance and clutter discrimination) filters.

For any practical application scenario the number of filters is large, e.g., 1,000 or more. Thus the problem of selecting the best filter subset from the bank at a particular time is critical. This problem is not yet resolved even though current and projected device technology supports correlation rates of 100 - 1000 input-filter pairs/sec. The "filter strategy" control problem limits practical designs because of the implied workload for the electronic control system.

A salient advantage of the correlation approach is its inherent shift-invariant response: the target need not be centered in the correlator input, and the location of the correlation peak in the output plane provides a location estimate for the target in the input plane.

Neural network approaches clearly have the capability to perform the ATR function, but their practical application is complexity-limited. From an information theory perspective, the ATR problem can be viewed as a very complex input-output relationship that must be learned with sufficient accuracy to provide statistically acceptable target-nontarget discrimination. Statistical performance is emphasized here because of the robust nature of realistic ATR scenes--a training set can at best be only statistically representative of the world of all possible input scenes. So far this problem description sounds well suited for neural network approaches and, in principle, these approaches are ideal. However, ATR performance must be invariant to target shifts, and when the additional complexity associated with this invariance is added to that associated with robust sets of input scenes, the resulting overall complexity exceeds levels that can be practically handled with current or near-term computational resources, i.e., neural network training time is unacceptably long, and the size of the network is too large either for simulation or for practical embodiment in a real-time system.

This paper concentrates on neural network approaches that augment the HAC concept. Three areas are considered:

1. Estimating target distortions relative to reference views.

2. Synthesizing new filters to follow dynamic target distortions.

3. Estimating confidence levels associated with correlation peaks to improve target-nontarget discrimination.

Work performed in all three areas is reviewed in the context of BPOF and TPAF correlation within the HAC concept. Two types of neural networks have been used in this work, "standard" backpropagation and "new" locally linear and stretch and hammer architectures developed at the University of Dayton. Background on BPOF and TPAF correlation is presented below and is followed by a review of pertinent neural network theory. Investigations in the three areas listed above are then discussed in turn.

2. CORRELATION WITH BPOF AND TPAF FILTERS

A recent review of optical correlation techniques was provided by Flannery and Horner [1989 (a)]. A renewed surge of interest in optical correlation for practical applications has been spurred by the recent development of the phase-only filter (POF) concept [Horner,1984], rapidly followed by the development of discrete-modulation-level BPOF and TPAF filters that support effective real-time implementation with currently available spatial light modulators (SLM) [Ross, 1983; Psaltis, 1984; Flannery, 1986]. These developments have made the HAC (Figure 1) a practical concept capable of implementation using current technology.

The BPOF is defined with two phase modulation levels (usually 0 and 180 degrees, corresponding to amplitudes of -1 and 1). It can be implemented with magneto-optic

SLMs (MOSLM [Ross, 1983; Davis,1989]), ferro-electric liquid crystal (FLC) SLMs [Johnson, 1990], and deformable mirror device (DMD) SLMs [Florence, 1990].

The TPAF may be viewed as an important extension of the BPOF that includes one additional modulation level: zero (i.e., the signal light is blocked at the filter positions or pixels having this level). In practice even the BPOF has zero-modulation levels due to its limited region of support (e.g.,finite SLM aperture). In the TPAF, elements are set to zero anywhere in the filter region according to an algorithm that provides (primarily) the benefit of improved target-nontarget discrimination. This improvement is achieved by blocking spatial frequencies where nontargets are expected to have relatively large spectral content [Flannery,1988 (a)].

The TPAF may be implemented in MOSLM devices using appropriate drive techniques to access a third "mixed-magnetization" state [Kast,1989]. Theoretical and experimental results amply demonstrate the improved discrimination (relative to BPOF correlation) provided by the TPAF [Lindell,1990;Flannery,1990;Flannery,1991]. Recent reports indicate good potential for implementing TPAF modulation in other SLM devices [Juday, 1991]. Another approach is to cascade a binary amplitude SLM with a phase-modulating SLM, which is less attractive from an optical engineering perspective.

All the limited-modulation filters discussed above (POF, BPOF, and TPAF) provide, in general, excellent correlation performance as characterized by sharper correlation peaks, greater peak intensity (correlation efficiency), and improved nontarget discrimination compared to the classic matched filter (which uses full complex amplitude modulation).

The signal-to-noise performance of limited-modulation filters has been a subject of great interest since intuition suggests that a price must be paid for restricting modulation levels. The classic matched filter by definition provides the best SNR (signal-to-noise ratio) for the case of additive Gaussian noise. However, the limited-modulation filters have shown generally superior discrimination against actual scene clutter [Horner, 1990]. Analysis is complicated by the lack of accepted standardized analytical models of practical clutter. Thus SNR or discrimination performance is scene- dependent, and many results are either anecdotal or suspect because of the limited robustness of test sets. Several theoretical treatments (limited by the assumption of Gaussian white noise) have been reported [Dickey, 1988, 1989; Kumar,1989,] but are not reviewed here. However, limited-modulation level filters exhibit typical SNR reductions of 3 to 10 dB (decibels) relative to matched filters for the white noise case but frequently provide better discrimination against practical noise patterns. Worth noting is a recent analytical treatment that derives tight bounds for SNR degradations for various limited-modulation filters [Farn,1990]. A summary of the current situation is that the presumed SNR penalties for BPOF and TPAF correlation are far outweighed by their advantages, which include the overwhelming advantage of practical implementation with available SLM devices. An additional practical advantage is the reduced amount of storage required for BPOF and TPAF filters (e.g., one or two binary bits per pixel) relative to complex-valued filters (e.g., at least 8 bits per pixel).

Effective techniques for formulating smart BPOF and TPAF filters have been developed and demonstrated in both simulations and experiments. A major design component of these filters is the specification of the zero-state pattern that optimizes SNR and/or other metrics, including peak intensity (or correlation efficiency) [Flannery,1990,1991;Kumar,1991]. Distortion invariance is addressed by adjusting the training-set weights of a composite in-class (target) image [Jared,1989(a); Flannery 1990,1991] or by direct iteration of the phase values of a POF pattern [Kallman,1987]. The threshold line angle (TLA) [Flannery,1988 (b);Farn,1988] is another design element of BPOF and TPAF formulations. It determines the relative weighting of odd and even symmetry components of the target image in the filter response. Small but significant improvements result from choosing an optimum TLA value.

A very recent and important development in TPAF optimization is formulation algorithms that address mixed-metric optimizations [Kuman, 1991; Flannery, 1991], i.e., compromises that simultaneously address two or more correlation performance metrics. These algorithms are motivated by the observation that filters optimized with regard only to SNR frequently have undesirable performance characteristics in other respects, including correlation efficiency so low as to preclude practical implementation and broad correlation peaks that hamper location-estimate post-processing. These undesirable characteristics are associated with very small support regions that result from optimizing only for SNR, i.e., most of the filter area is set to the zero-modulation state.

Results are presented here from a recent case study of TPAF smart filter formulation using mixed metrics [Flannery,1991] to illustrate the state of the art in this area. Figure 2 provides an example input scene of a tank on a bushy background and the binary version of the scene as used in the study (to match the binary input capability of the experimental correlator that uses MOSLM devices at both input and filter planes). Smart TPAF filters were formulated to cover 20 degrees of in-plane target rotation and 12% of in-plane target scale variation while rejecting the background shown as well as three other backgrounds of diverse character. When the filter was optimized only for SNR (i.e., background discrimination) it provided excellent discrimination (averaging 8.75 dB over the four backgrounds), but the correlation efficiency was so low that practical implementation in the experimental correlator was marginal. With optimization to a mixed metric reflecting equal emphasis on both SNR and peak intensity, the filter still provided excellent background discrimination (averaging 7.62 dB) and also provided over twice the peak intensity of the previous filter, which was a practical level for the experimental correlator. Information generated during filter design indicated that in this case the compromise filter provided 85% of the SNR of the best-SNR design and 85% of the peak intensity of the filter designed for maximum peak response. Figure 3 shows simulated and experimental correlation intensity plots for the compromise filter with the input of Figure 2(b).

3. NEURAL NETWORK ARCHITECTURES

This section reviews neural network architectures and training algorithms used in the work discussed here. Backpropagation (e.g., Lippmann, 1987), locally linear [Gustafson, 1991b], and stretch and hammer [Gustafson, 1991a] neural networks are considered.

Backpropagation-trained neural network:

Feedforward neural networks with weights determined by the backpropagation training are the most commonly used neural networks in engineering applications. It has been shown in principle that with three neuron layers and with enough neurons in the hidden layer (the layer not directly connected to either inputs or outputs) any arbitrary input-output relationship may be learned. The backpropagation algorithm is equivalent to a gradient descent optimization in weight space with a least-square-error goal. However, no general prescription for the various design elements is available; and judicious initial choices and trial-and-error design iterations are the normal methods for designing networks for particular applications. Accuracy of learning, interpolation, and extrapolation relative to training set data are important performance parameters. These parameters are affected by training set selection, training schedule, RMS error goal, and network topography. A large body of common experience relating these elements has evolved, and the results reported here were obtained using conventional methods based on this experience.

Locally linear neural network:

As noted above, multilayer feedforward neural networks that use backpropagation or similar training algorithms are by far the most common in engineering applications. However, these neural networks have several limitations. First, they generally lack the coordinate invariance property, according to which the testing output is unchanged if the testing and training (or data) inputs are translated, rotated, or scaled. Second, without excessive training they generally lack the data interpolation property, according to which the testing output is the training output if the testing inputs are the training inputs [Poggio and Girosi, 1990]. Third, they generally lack the linear representation property, according to which any testing point is on a linear surface if all training points are on this surface. Finally, they generally lack the data bootstrapping property, according to which the testing outputs have least squared error for any training points that are transformed into testing points. In contrast, locally linear neural networks have all of the above desirable characteristics. There are two steps in training: transforming the inputs to invariant coordinates and finding a plane through each training point that satisfies a bootstrapping property in that the plane predicts, with minimum squared error, a specified number of training point nearest neighbors. In testing the plane through the training point nearest to the testing point (in the transformed inputs) is used to find the testing point output. Locally linear neural networks extend nearest neighbor techniques because in training they find a plane (having as many dimensions as there are inputs) through each training point that in general has non-zero slope.

Stretch and hammer neural network:

Stretch and hammer neural networks also avoid the many limitations of backpropagation training and use radial basis function methods to achieve advantages in generalizing training examples. These advantages include (1) exact learning, (2) maximally smooth modeling of Gaussian deviations from linear relationships, (3) identical outputs for arbitrary linear combination of inputs, and (4) training without adjustable parameters in a predeterminable number of steps. Stretch and hammer neural networks are feedforward architectures that have separate hidden neuron layers for stretching and hammering in accordance with an easily visualized physical model. Training consists of (1) transforming the inputs to principal component coordinates, (2) finding the least squares hyperplane through the training points, (3) finding the Gaussian radial basis function variances at the column diagonal dominance limit, and (4) finding the Gaussian radial basis function coefficients. The Gaussian radial basis function variances are chosen to be as large as possible consistent with maintaining diagonal dominance for the simultaneous linear equations that must be solved to obtain the basis function coefficients. This choice insures that training example generalization is maximally smooth consistent with unique training in a predeterminable number of steps.

4. DISTORTION PARAMETER ESTIMATION USING NEURAL NETWORKS

The problem addressed by the work reported in this section is as follows. Given a starting condition consisting of correlation of an input target object with a matching filter, estimate distortion parameter(s) (e.g., rotation angle and scale factor) as the input object is distorted from the original view. The value of distortion parameter estimates for use in a filter control strategy for a HAC is obvious.

The approach investigated for distortion parameter estimation in one study [Gustafson,1990] involved sampling the shape of the correlation peak (the desired peak in response to the target) to derive inputs for a backpropagation-trained neural network. In-plane target rotation was used as the distortion mechanism and thus the goal of the network was to estimate the rotation angle relative to the initial view. A binary image of an aircraft was used as the target, and 128x128 sample correlation simulations were performed. Intensity samples were taken on a 5x5 pixel grid centered on the correlation peak. These samples were normalized to the central value, thus defining 24 inputs to the neural network. The network was trained for 5-degree intervals of target rotation with the filter held constant at the initial view. The neural network estimated in-plane rotation to within +/-5 degrees over a range of -40 to +40 degrees from the initial view.

Another study [Gustafson,1991] applied the same approach to an IR truck target image and used both backpropagation and locally linear neural networks. The backpropagation network estimated rotation angle with errors less than 2 degrees, whereas the LLN provided errors less than 0.35 degrees (see Figures 4 and 5). Estimation was seriously degraded when realistic backgrounds were introduced in the input scenes, presumably due to the resulting distortion of the correlation peak relative

to the zero-background case. Training using background-distorted inputs did not reduce this degradation to an acceptable level.

To reduce the degradation caused by backgrounds, a new sampling approach, input-plane sampling, was introduced. In this approach (successful) correlation is used to establish a reference point (location estimate) for sampling the target intensity in the input plane. Sampling was again based on a 5x5-pixel grid, but in general the sampling pattern was chosen to keep most or all samples on the target image (thus excluding background effects). A study using the same truck target showed that rotation angle could be predicted over the full rotation range of 360 degrees with errors well below two degrees even in the presence of clutter backgrounds (see Figure 6) [Olczak,1991]

Although perhaps not obvious, a boot-strapping assumption is implicit in the above distortion-estimation approaches: good correlation must be maintained at all times. The correctly-located correlation peak must be detectable (i.e., must exhibit a sufficiently high peak-to-clutter ratio in the correlation plane) to enable proper location of the sampling grid for either approach discussed above. The input-plane sampling approach is generally superior because it provides samples that are less corrupted in cases where there is a slight mis-match between filter and target but where good correlation is still obtained.

Distortion parameter estimation using neural network approaches shows promise for aiding HAC filter control strategy in tracking modes of operation (where successful correlation has been achieved and must be maintained through rapid target evolution). Current research is investigating issues such as sensitivity to sampling grid location accuracy, optimization of sampling grid patterns, and extension to multiple distortion dimensions.

5. FILTER SYNTHESIS USING NEURAL NETWORK APPROACHES

This section reviews efforts that investigate the modification or synthesis of TPAF filters using neural network approaches.

Filter amplitude states:

In an initial simulation study the bandpass of a BPOF was covered by four binary (on-off) amplitude control rings driven by four neural network outputs [Flannery,1989 (b)]. The inputs were the integrated power spectral densities in the input scene taken over the same four spectral rings (readily available on a real-time basis in an optical correlator system). The goal was to maximize correlation signal-to-noise for a target imbedded in different noise samples by optimum control of the correlator bandpass. This control may be viewed as setting zero-states in a TPAF on a very coarse framework. The simulations were successful in that the neural network was easily trained to provide near-optimum bandpass configurations for a variety of input noise conditions.

Filter phase states:

Two basic issues must be resolved to define a general filter synthesis approach: (1) A filter representation or parameterization must be used that is consistent with the number of available outputs in current neural network practice (which is limited in both simulations and hardware implementations) and (2) the inputs to the network must be specified.

To illustrate the first issue, consider the complete specification of a 128x128-pixel TPAF. In excess of 32,000 binary values must be determined, whereas the neural network computation resources typically used support at most 600 outputs. Thus ways to reduce the number of parameters used to represent a filter must be developed. One approach is to impose a bandpass limit on the filter (one quarter of the full Nyquist bandwidth) and to consider only BPOF filters with cosine (even) symmetry. Another approach is to extend the bandpass to almost the full Nyquist interval by grouping pixels in 3x3 super-pixels (called "nonapixels" hereafter), each controlled by a single modulation value. The first approach severely smooths the filter impulse response, whereas the second approach limits the extent of the impulse response to about one third of the input field of view.

The second issue, defining network inputs, may be addressed with the same two sampling approaches already discussed in the context of distortion parameter estimation (correlation peak shape and input-plane sampling).

Steps taken during the course of investigations on filter synthesis at the University of Dayton are summarized here. These investigations paralleled the distortion-estimation investigations in the sense that input-plane sampling was found to be much superior to sampling correlation peak. The quarter-Nyquist bandpass filters were successfully synthesized but were of little practical value due to their limited bandpass; they exhibited insufficient discrimination against background clutter and non-target vehicles (see Figure 7). Nonapixel filters were also successfully synthesized and performed well [Olczak, 1991]. For nonapixel filters the target extent was less than 40 pixels on a 128-pixel format, thus satisfying the constraint mentioned above.

Recent previously unreported results on the synthesis of nonapixel BPOFs using input-plane sampling are shown in Figures 8, 9 and 10. Figure 8(a) is a typical input image showing the target truck and another vehicle superimposed on a clutter background. Figure 8(b) is a nonapixel filter pattern for the truck. Figures 8(c) is a correlation intensity pattern from a correlation simulation using this input and filter. Figure 9 and 10 provides plots of filter peak-to-clutter performance over 360 degrees of target rotation for synthesized nonapixel filters using backpropagation and stretch and hammer neural networks, respectively. Data for two other filters also are plotted for reference in Figure 9; the simple (single-view) BPOF and the best possible BPOF that can be designed at each rotation angle. As is apparent in the plots, the performance of the neural-synthesized filters approaches that of the best-possible filters to a satisfying degree. These plots involve the target superimposed on one of several clutter backgrounds; results with the other backgrounds were similar.

The results seem impressive because the neural network provides filter information equivalent to about 45 distinct single-view filters, which would require about 3.5 kB (kilobytes) of digital storage (assuming about 600 binary bits/filter). Neural network filter synthesis also provides an implicit control mechanism because no indexing or searching through a filter database is required. The only caveat is the same one discussed in the review of distortion-estimation techniques, which is that this approach is based on maintaining good correlation in a dynamic scenario (i.e., successfully tracking the target) so that the location estimate derived from the correlation peak may be used to accurately center the input-plane sampling grid.

An important question concerns the affect of target distortions not learned by the network, e.g., scale changes. More generally, how can neural network filter synthesis be extended to distortions involving two or more degrees of freedom, e.g., azimuth, elevation, scale, etc.? The associated complexity growth may tend to drive the problem outside bounds practically addressable by available neural network resources. This and other issues similar to those mentioned for distortion estimation are under investigation, as are techniques for neural network synthesis of the zero-modulation pattern required to form a full TPAF.

6. CORRELATION CONFIDENCE-LEVEL ESTIMATION USING NEURAL NETWORK APPROACHES

A correlation involving a target surrounded by clutter will result in a peak corresponding to the target and other (hopefully smaller) peaks in response to clutter. Normally the desired peak must exceed other peaks by some margin (e.g., 3 dB) for correlation to be useful. If the clutter level in the input scene is gradually increased, a point is reached where the filter is no longer useful by this standard. If the filter is a distortion-invariant smart filter, it might be possible to substitute a more target-specific filter which would furnish better discrimination. This approach is undesirable in general because it implies a large storage bank of "more-specific" filters and because there is an implied control problem (i.e., which of the many more-specified filters corresponding to a single smart filter should be substituted?). Neural network techniques that estimate confidence levels for correlation peaks are potential approaches to this problem. Note that if the original filter was already of the more-specific variety, some augmentation of the correlation process is mandatory if useful results are to be obtained.

The same two sampling techniques, correlation peak and input-plane, were investigated as neural network inputs, and again input-plane sampling proved superior. Recent previously unreported results are synopsized here to illustrate the potential of this approach.

A set of terrain board images was provided by Martin Marietta, Strategic Systems, Denver, Colorado. These 128x128-pixel images included three vehicle targets on a cluttered background. The image set spanned elevations of 15 to 45 degrees and azimuths of 0 to 90 degrees. A matrix of 45 images covering this two-dimensional distortion range was used for this study, including 30 for training and 15 for testing. All

correlations were performed with a single BPOF constructed for the central view in the image matrix. Thus correlations were poor in response to the target for many of the input images. For each correlation four correlation peaks were considered: one for each of the three vehicles (including the target vehicle) and the highest peak in response to input clutter.

The input-plane sampling mask consisted of eight spokes of even angular distribution, each 16 pixels long. These 128 samples were augmented by 8 inputs derived by applying a one-dimensional Roberts edge-location operator along each spoke and finding the location of the strongest response. The network was trained to provide two complementary-logic outputs indicating whether the sampled object was a target or not. These outputs were algebraically combined to yield a confidence level. Figure 11 is a histogram plot of the results of the trained network applied to the 15-image (60-peak) test set. The a-priori target-to-nontarget ratio for these inputs is 1:3. As can be seen, the network provided excellent separation of targets and nontargets (i.e., very useful confidence level outputs). Figure 12 shows correlation intensity plots corresponding to perfect target-filter match (part a) and extreme target-filter mismatch (part b). Other correlations are expected to fall between these extremes.

Current work is addressing more challenging images, variations of the input-plane sampling mask, and the application of different types of neural network architectures to this problem.

7. CONCLUSION

The work reviewed here has shown definite promise for the development of neural network approaches that augment hybrid adaptive optical correlation systems. Although other approaches may be defined for the use of neural networks in automatic target recognition, the approaches discussed here involve an advantageous combination of the strengths of the two underlying technologies. In particular, these approaches allow the two basic strengths of optical correlation (shape-dependent discrimination and intrinsic location estimation) to be used to their full extent. Neural network augmentation techniques, when incorporated with the HAC concept, should permit the development of more efficient and powerful systems for addressing complex pattern recognition.

8. REFERENCES

J. Davis and J. Waas, "Current Status of the Magneto-optic Spatial Light Modulator," Proc. SPIE, Vol. 1150, pp. 27-43, August, 1989.

F. Dickey, K. Stalker, and J. Mason, "Bandwidth considerations for binary phase-only filters," Applied Optics, Vol. 27, pp. 3811-3818, 15 September 1988.

F. M. Dickey, L.A. Romero, "Dual Optimality of Phase-Only Filter," Optics Letters, Vol. 14, pp. 4-5, 1 Jan. 1989.

M.W. Farn and J.W. Goodman,"Optimal binary phase-only matched filters," Applied Optics, Vol. 27, pp. 4431-4437, November 1, 1988.

M. Farn and J. Goodman, "Bounds on The Performance of Continuous and Quantized Phase-only Matched Filters," J. Opt. Soc. Am. A, Vol. 7, pp. 66-72, January 1990.

K. Fielding and J. Horner, "Clutter Effects on Optical Correlators," Proc. SPIE, Vol. 1151, pp. 130-137, August, 1990.

D. Flannery, A. Biernacki, J. Loomis, and S. Cartwright, "Real-time Coherent Correlator Using Binary Magnetooptic Spatial Light Modulators at Input and Fourier Planes," Applied Optics, Vol. 25, pp. 466, 1 February 1986.

D. Flannery, J. Loomis, and M. Milkovich, "Transform-ratio ternary phase-amplitude filter formulation for improved correlation discrimination," Applied Optics, Vol. 27, pp. 4079-83, 1 October 1988. (a)

D. Flannery, J. Loomis, and M. Milkovich, "Design elements of binary phase-only correlation filters," Applied Optics, Vol. 27, pp. 4231-4235, 15 October 1988. (b)

D. L. Flannery and J. L. Horner, "Fourier Optical Signal Processors," Proc. IEEE, Vol. 77, pp. 1511-1527, October 1989. (a)

D. Flannery, S. Gustafson, and D. Simon, "On-line Adaptation of Optical Correlation Filters," Proc. Aerospace Applications of Artificial Intelligence," Dayton, Ohio, October 1989. (b)

D. Flannery, W. Hahn, and E. Washwell, "Application of Ternary Phase-amplitude Correlation Filters to LADAR Range Images," Proc. SPIE, Vol. 1297, pp. 194-206, April, 1990.

D. Flannery, W. Phillips, and R. Reel, "A Case Study of Design Trade-offs for Ternary Phase-amplitude Filters," Proc. SPIE, Vol. 1564 (in press), San Diego, California, July, 1991.

J. Florence and R. Juday, "Full Complex Spatial Filtering with a Phase-mostly DMD," Proc. SPIE, Vol. 1558 (in press), San Diego, CA, July 1991.

S. Gustafson, D. Flannery, and D. Simon, "Neural Networks with Optical Correlation Inputs for Recognizing Rotated Targets," Proc. SPIE, Vol. 1294, pp. 171-179, Orlando, Florida, April 1990.

S. C. Gustafson, G. R. Little, J. S. Loomis, and T. S. Puterbaugh, "Stretching and Hammering Algorithm for Data Representation and Generalization," submitted to Comm. in Appl. Numerical Meth., March 1991 (a).

S. Gustafson, G. Little, and E. Olczak, "Locally Linear Neural Networks for Optical Correlators," Proc. SPIE, Vol. [1469], pp. 268-274, Orlando, Florida, April 1991 (b).

J.L. Horner, J.R. Leger, "Pattern Recognition with Binary Phase-Only Filters," Applied Optics, pp. 609-611, Vol. 24, 1 March 1985.

D. Jared and D. Ennis, "Inclusion of filter modulation in synthetic-discriminant function filters," Applied Optics, Vol. 28, pp. 232-239, 15 January 1989. (a)

D. Jared and K. Johnson, "Ferroelectric Liquid Crystal Spatial Light Modulators," Proc. SPIE, Vol. 1150, pp. 46-60, August 1989. (b)

R. Kallman, "Direct construction of phase-only correlation filters," Proc. SPIE, Vol. 827, pp. 184-190, August, 1987.

B. Kast, M. Giles, S. Lindell, and D. Flannery, "Implementation of ternary phase-amplitude filters using a magneto-optic spatial light modulator," Applied Optics, Vol. 28, pp. 1044-46, 15 March 1989.

B. Kumar and Z. Bahri, "Phase-only filters with improved signal-to-noise ratio," Applied Optics, Vol. 28, pp. 250-257, 15 January 1989. (a)

B. Kumar and Z. Bahri, "Efficient algorithm for designing a ternary valued filter yielding maximum signal to noise ratio," Applied Optics, Vol. 28, pp. 1919-1925, 15 May 1989. (b)

B. Vijaya Kumar, "Effects of Quantizing the Phase in Correlation Filters," proc. SPIE, Vol. 1151, pp. 166-173, August 1990.

B. Vijaya Kumar and C. Hendrix, "Design and Testing of 3-level Optimal Correlation Filters," Proc. SPIE, Vol. 1564 (in press), San Diego, CA, July 1991.

B. Vijaya Kumar and R. Juday, "Design of phase-only, binary phase-only, and complex ternary matched filters with increased signal-to-noise ratios for colored noise," Optics Letters, Vol. 16, pp. 1025-1027, 1 July 1991.

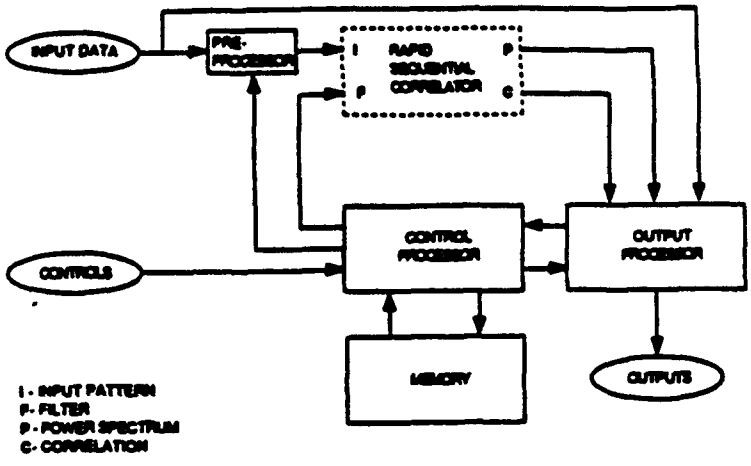
S. Lindell and D. Flannery, "Experimental Investigation of Transform Ratio Ternary Phase-amplitude Filters for Improved Correlation Discrimination," Optical Engineering, Vol. 29, pp. 1044-1051, September 1990.

R. P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP Mag., Vol. 4, pp. 4-22, 1987.

E. Olczak, "Neural Networks for The Hybrid Adaptive Correlator," Thesis submitted to University of Dayton for Master of Science Degree in Electro-optics, April 1991.

D. Psaltis, E. Paek, and S. Venkatesh, "Optical Image Correlation with a Binary Spatial Light Modulator," Optical Engineering, pp. 698-704, Nov./Dec. 1984.

W. Ross, D. Psaltis, and R. Anderson, "Two Dimensional Magneto-optic Spatial Light Modulator for Signal Processing," Optical Engineering, Vol. 24, pp. 485-490, 1983.



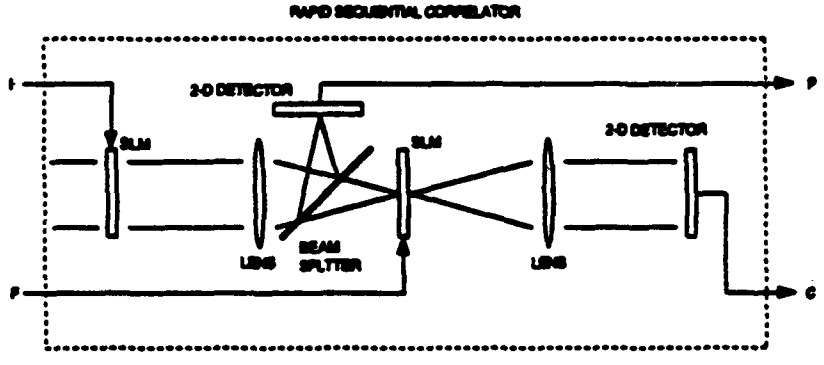
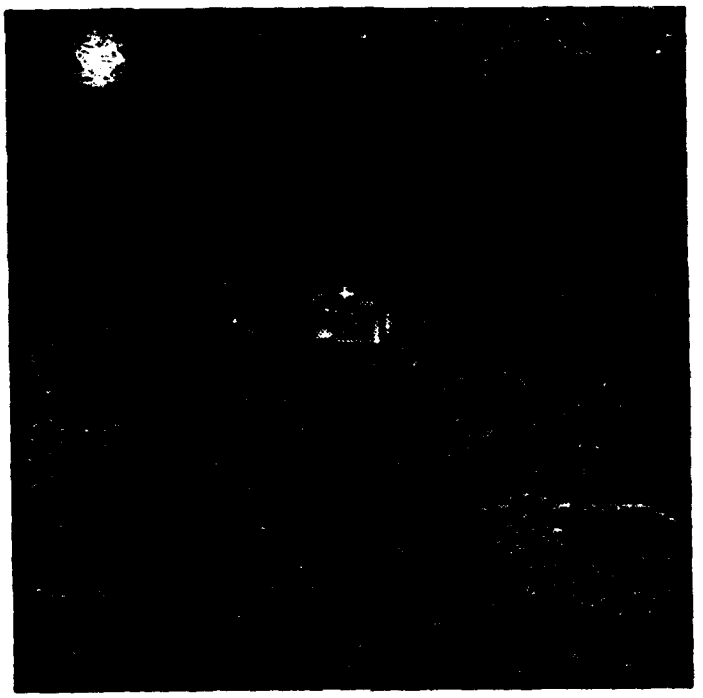


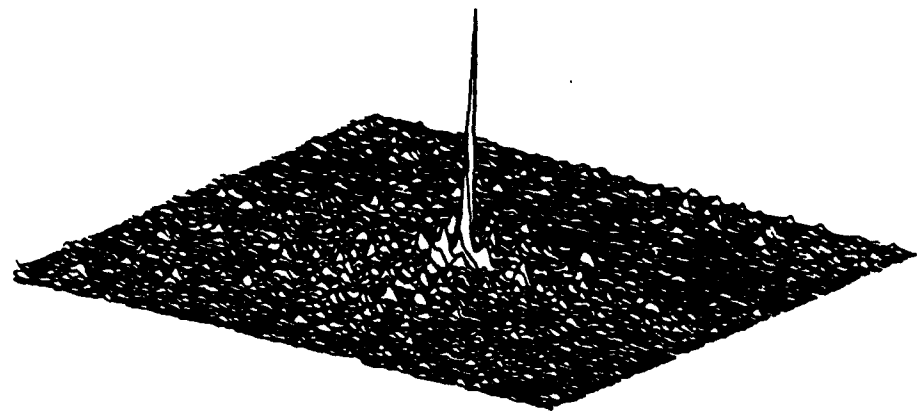
Fig. 1. Hybrid adaptive correlator (HAC) concept.



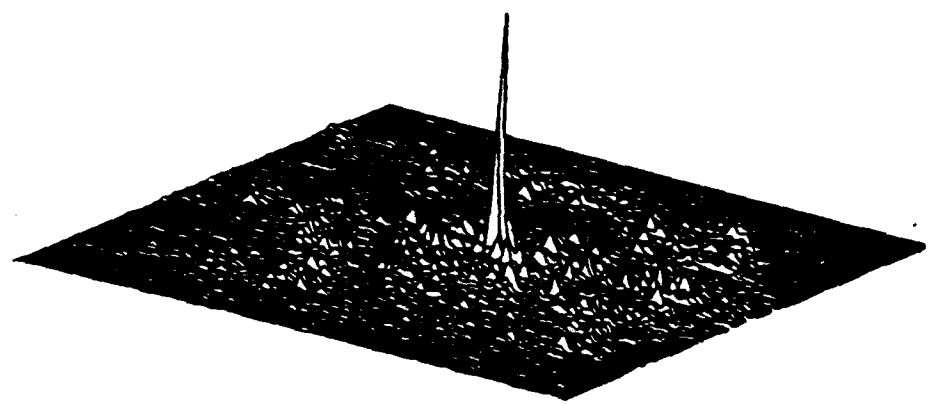
(a)

(b)

Fig. 2. Gray scale (a) and binary (b) versions of example target-on-background scene.



(a)



(b)

Fig. 3. Simulated (a) and experimental (b) correlation intensity for compromise filter.

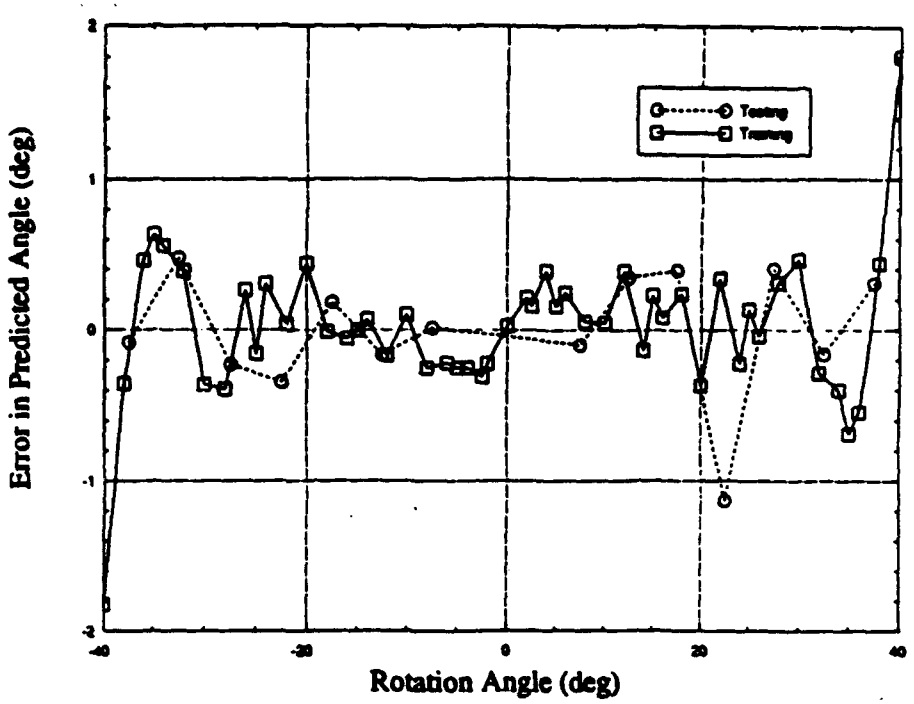


Fig. 4. Rotation prediction using backpropagation neural network with correlation peak inputs.

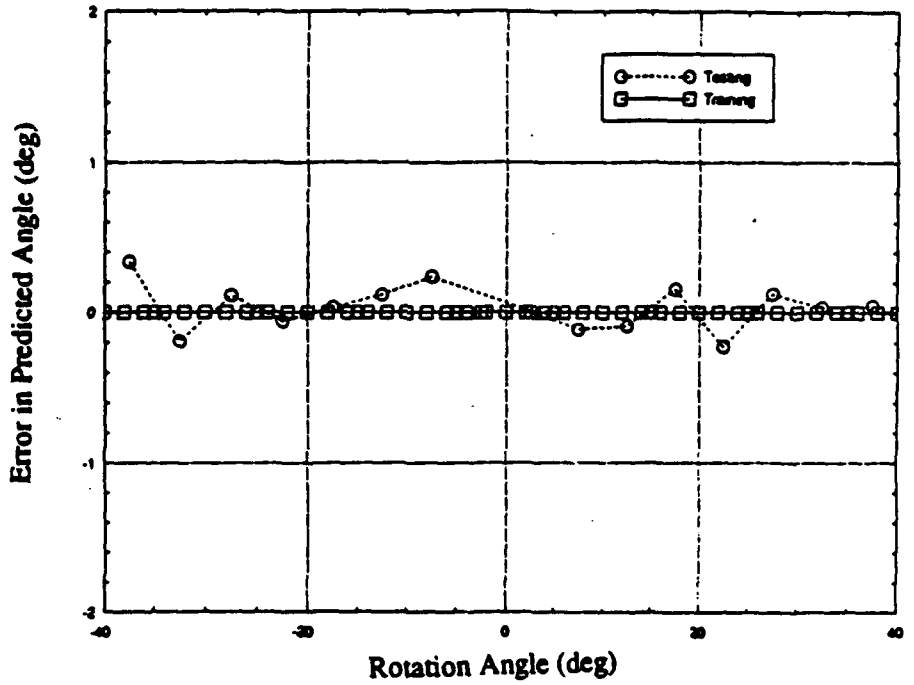


Fig. 5. Rotation prediction using locally linear neural network with correlation peak inputs.

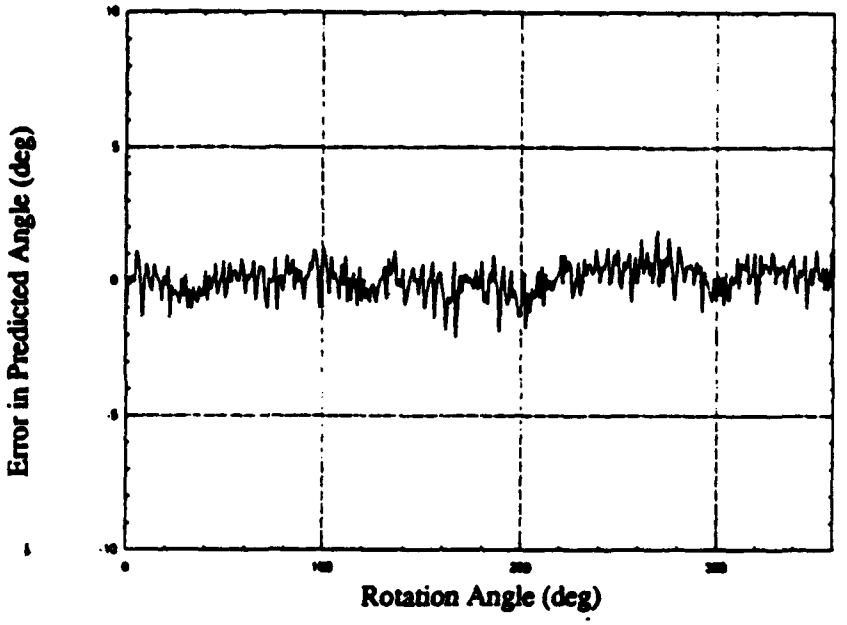


Fig. 6. Rotation prediction using backpropagation neural network with input image inputs.

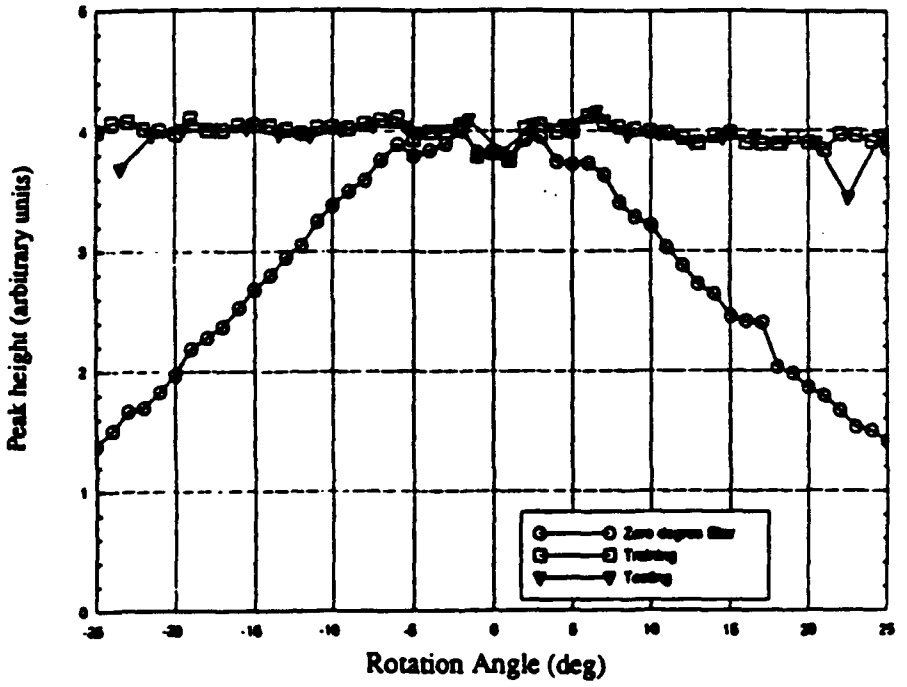


Fig. 7. Peak height performance of every-pixel filter synthesized by backpropagation neural network.

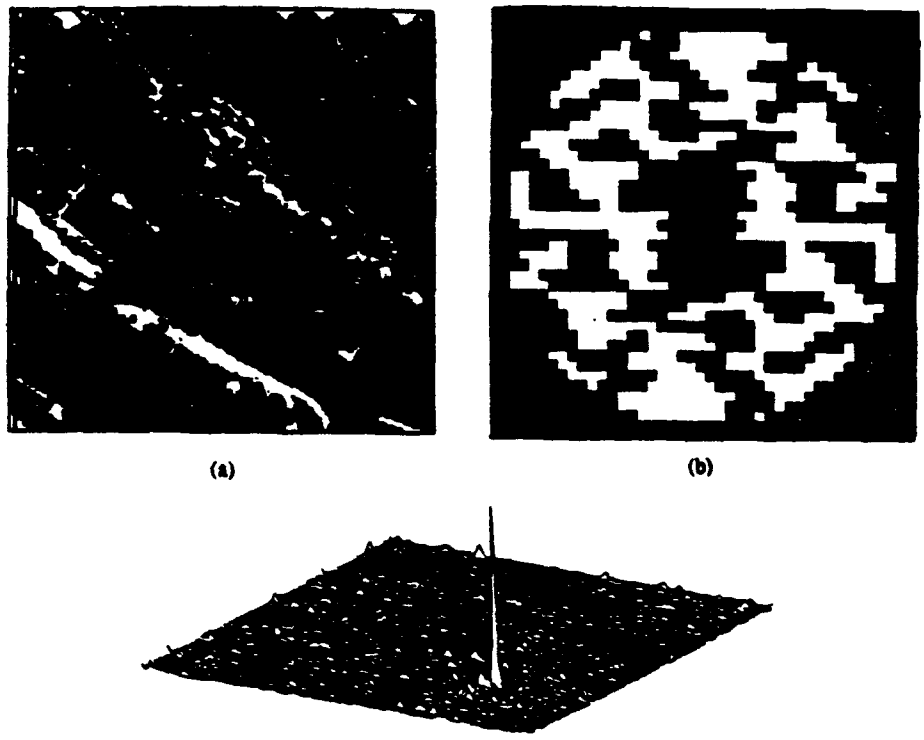


Fig. 8. Typical input image (a), corresponding neural-network-synthesized nonapixel filter (b), and resulting correlation intensity (c).

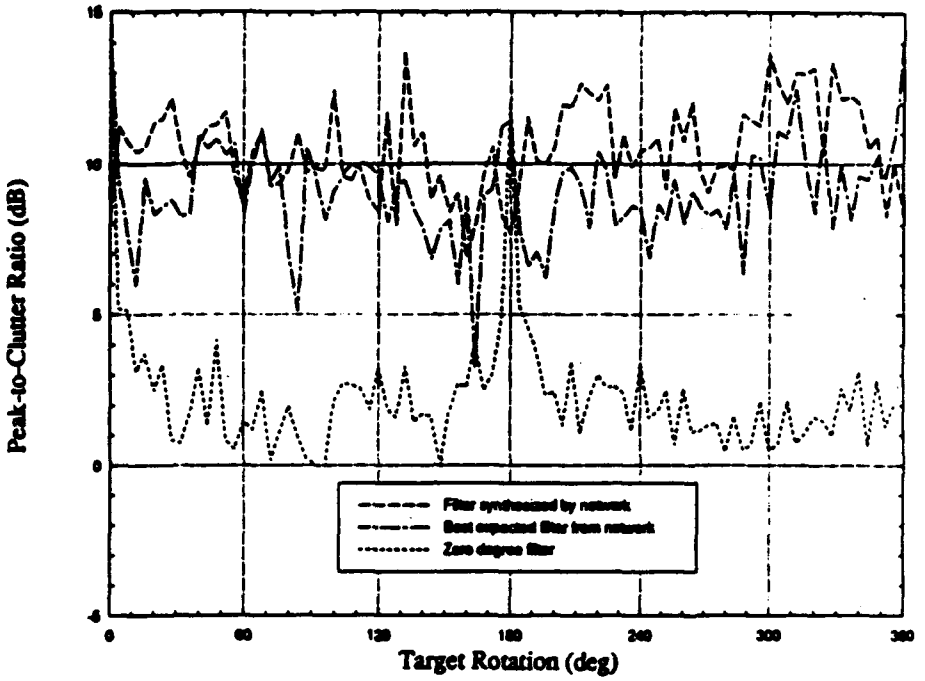


Fig. 9. Peak-to-clutter performance of nonapixel filter synthesized by backpropagation neural network.

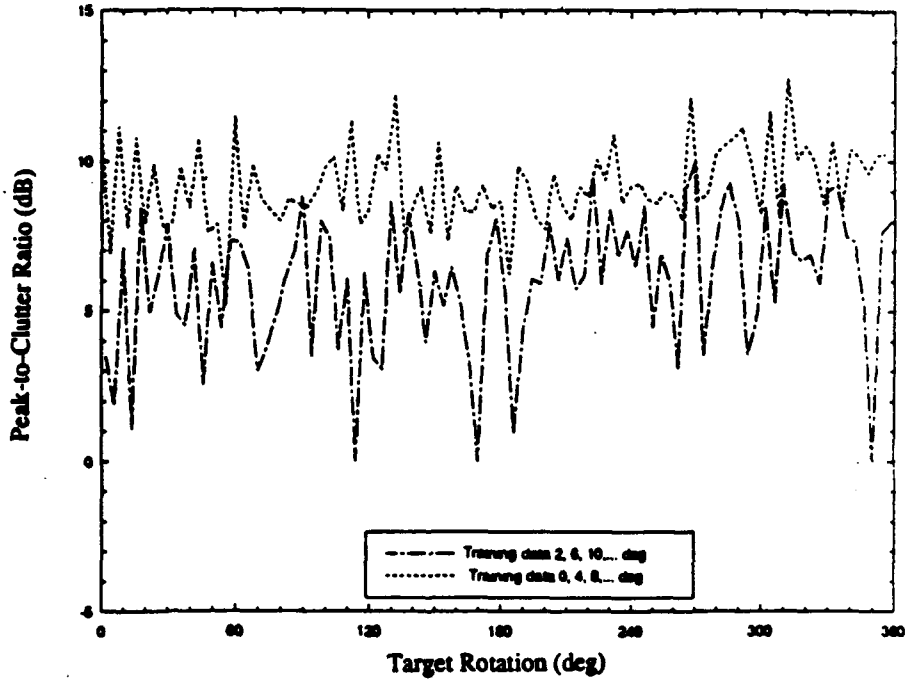


Fig. 10. Peak-to-clutter performance of nonapixel filter synthesized by stretch and hammer neural network.

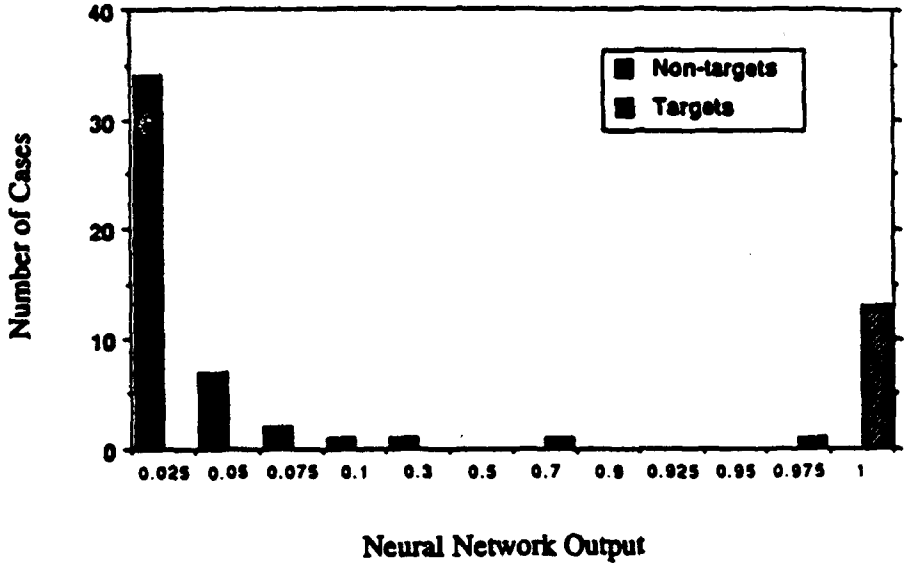
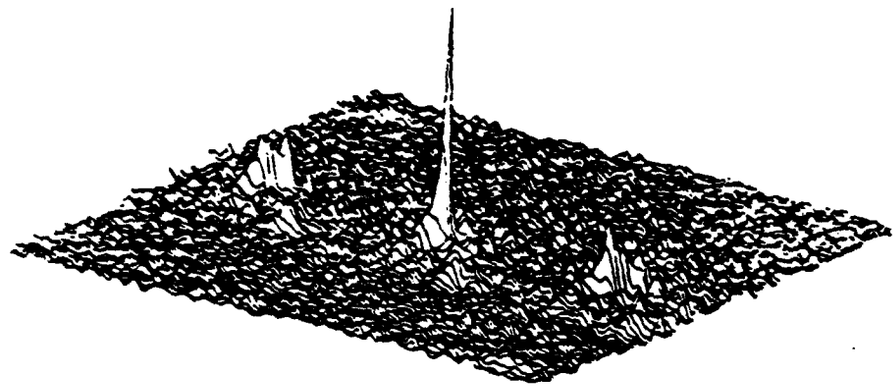


Fig. 11. Histogram of neural network target detection.



(a)

(b)

Fig. 12. Correlation intensity for target-filter perfect match (a) and extreme mismatch (b).

SPIE Vol. 1710 Science of Artificial Neural Networks (1992)

STRETCH AND HAMMER NEURAL NETWORKS

**Steven C. Gustafson, Gordon R. Little,
Mark A. Manzardo, and Todd S. Puterbaugh***

**Research Institute, University of Dayton
300 College Park, Dayton, OH 45469-0140**

***Current address: Science Applications International Corporation.,
101 Woodman Dr., Dayton, OH 45431**

Abstract

Stretch and hammer neural networks use radial basis function methods to achieve advantages in generalizing training examples. These advantages include (1) exact learning, (2) maximally smooth modeling of Gaussian deviations from linear relationships, (3) identical outputs for arbitrary linear combination of inputs, and (4) training without adjustable parameters in a predeterminable number of steps. Stretch and hammer neural networks are feedforward architectures that have separate hidden neuron layers for stretching and hammering in accordance with an easily visualized physical model. Training consists of (1) transforming the inputs to principal component coordinates, (2) finding the least squares hyperplane through the training points, (3) finding the Gaussian radial basis function variances at the column diagonal dominance limit, and (4) finding the Gaussian radial basis function coefficients. The Gaussian radial basis function variances are chosen to be as large as possible consistent with maintaining diagonal dominance for the simultaneous linear equations that must be solved to obtain the basis function coefficients. This choice insures that training example generalization is maximally smooth consistent with unique training in a predeterminable number of steps. Stretch and hammer neural networks have been used successfully in several practical applications.

1. Physical Model

In the same sense that thin plate spline interpolation has a "bending" model in which an elastic plane is deformed into contact with the data points, stretch and hammer neural networks [Gustafson et al., 1991, 1992] have a physical model in which the data input plane is similarly deformed. In this model the input plane is first stretched along orthogonal coordinates located in the plane so that the data inputs (relative to their means) have equal variances and zero covariances. (This procedure is a principal components transformation on the data inputs). Next a least squares hyperplane is found for the transformed data. Finally, the data inputs in the stretched coordinates are projected onto the hyperplane and hammered into contact with the data outputs using numerous small strikes so that the hyperplane is

smoothly deformed. Hammering has (typically) Gaussian precision with variance such that the ratio of strike density at each data input to the sum of strike densities at all other inputs, i.e., the ratio of hits to misses, exceeds unity.

2. Advantages

It has been proved that if the number of training points is much greater than the number of inputs, the stretch and hammer neural network places guaranteed upper limits on required numerical precision and number of computational steps and that it also places guaranteed lower limits on certain measures of interpolation smoothness and stability [Gustafson et al. 1992]. Here smoothness as measured by the smallest hammering standard deviation is maximized by selecting the largest value consistent with an affordable level of computational effort, and this value is obtained by setting the minimum ratio of hits to misses equal to $(\kappa + 1)/(\kappa - 1)$, where κ is the largest acceptable 2-norm condition number of the matrix F whose inverse determines the basis function coefficients. Also, stability as measured by the reciprocal of the root mean square fractional change in the number of strikes at each data point is bounded by $s \geq [(\kappa r)^{-1} - 1]/2$, where r is the fractional root mean square change in the data outputs (i.e., good stability implies that small data output changes yield small interpolation function changes).

Since the stretch and hammer neural network is an interpolator and extrapolator (although modifications to enforce additional smoothness at the expense of exact data fitting are possible), exact learning is achieved. Also, since the hammering precision of radial basis functions is typically Gaussian with the maximum practical standard deviation, the network provides maximally smooth modeling of Gaussian deviations from linear relationships. Furthermore, since the data inputs are transformed by stretching to principal component coordinates, the network provides identical outputs for arbitrary linear combinations of inputs. Finally, training is achieved without adjustable parameters with a computational effort governed by κ in terms of bounds on required numerical precision and number of computational steps.

3. Training Procedure

Typical stretch and hammer neural network training consists of standard operations that yield the mathematical specification outlined in Figure 1 and detailed in the references [Gustafson et al. 1991, 1992]. First the data inputs x_i and their means $\bar{x}_i$ for all data points are expressed in principal component coordinates u_i where a_{ij} are the linear transformation coefficients (see Figure 1 for notation). Next a least squares hyperplane is fitted to the data points, where b_0 is the hyperplane intercept and $b_1, b_2, \ldots, b_n$ are the hyperplane slopes. Next the Gaussian radial basis function standard

deviations s_i are selected to be as large as possible consistent with maintaining acceptable diagonal dominance for F whose elements are given by the radial Gaussian functions f_i evaluated at the stretched data inputs. Thus, each s_i is selected such that $(1+d_j)/(1-d_j) = \kappa$, where d_j is the sum of the off-diagonal elements of the jth column of F. Finally, the basis function coefficients c_k are obtained by solving m simultaneous linear equations in m unknowns.

4. Example Results

Figures 2, 3, and 4 show example results for the stretch and hammer neural network. Figure 2 compares stretch and hammer and natural cubic spline interpolation for one input. Note that the curves are comparable except for sparse interpolation regions, where the stretch and hammer curve approaches the least squares line. Figure 3 shows a least squares plane fitted and hammered to four two-input training points with the two stretch (principal component) coordinates indicated, and Figure 4 shows these training points in the stretched coordinates with a least square plane fitted and hammered in these coordinates. Note that interpolation in the stretched coordinates is smoother than in the original coordinates.

5. Practical Application

Figures 5 and 6 show practical application of the stretch and hammer neural network to an adaptive optical correlation system designed to track targets in images [Flannery and Gustafson, 1991]. The network synthesized binary phase Fourier plane filters using 31 samples of the target region in the input scene. The network was trained to synthesize filters that maintain a high correlation peak to clutter ratio for clutter backgrounds not used in training (Figure 5) and for such backgrounds plus target rotation angles not used in training (Figure 6). Note the favorable comparison with the zero degree filter (a fixed filter designed for zero degree rotation) and the best expected filter (the best filter that could have been synthesized).

6. Acknowledgment

This research was supported in part by the Air Force Rome Laboratory on Contract F19628-91K-0014.

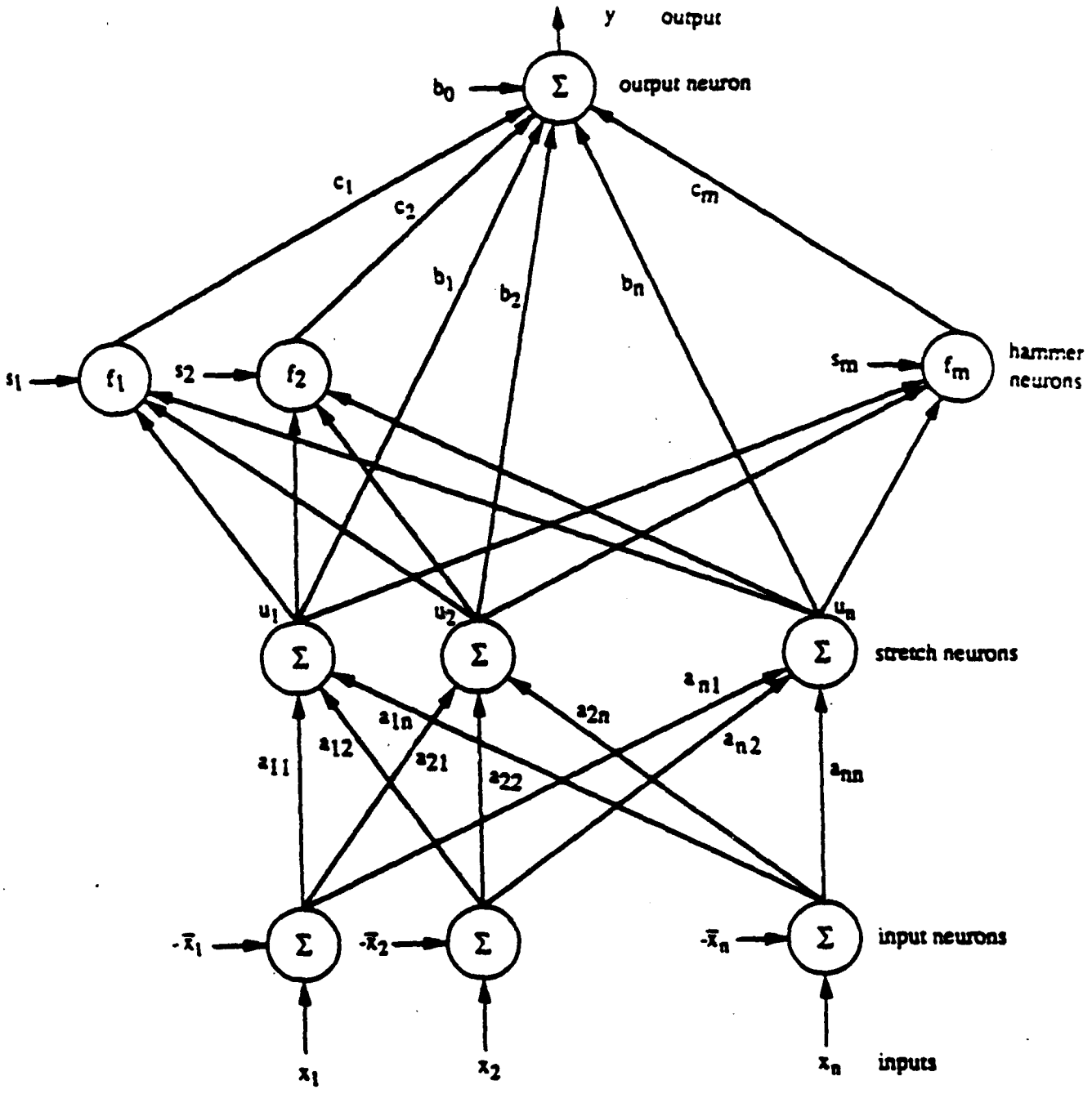
7. References

1. D. L. Flannery and S. C. Gustafson, "Adaptive optical correlation using neural network approaches," SPIE Critical Review, CR-40-02, San Jose, CA, 1991.

2. S. C. Gustafson, G. R. Little, and J. S. Loomis, "Generalization with Guaranteed Bounds on Computational Effort, Smoothness, and Stability," submitted to *IEEE Trans. Neural networks*, 1992.

3. S. C. Gustafson, G. R. Little, J. S. Loomis, T. S. Puterbaugh, and P. S. Raeth, "Stretch and Hammer Neural Network, *Invention Disclosure No. 116*, University of Dayton, 1991.

4. M. A. Manzardo, "Optical Filter Synthesis Using Artificial Neural Networks," M.S. Thesis, Univ. of Dayton, 1992.



$$y = b_0 + b_1u_1 + \ldots + b_nu_n + c_1f_1 + c_2f_2 + \ldots + c_mf_m$$

$$f_i = \exp[-(u_1^2 + u_2^2 + \ldots + u_n^2)/2s_i^2] \ , \ i = 1, 2, \ldots, m$$

$$u_j = a_{j1}(x_1 - \bar{x}_1) + a_{j2}(x_2 - \bar{x}_2) + \ldots + a_{jn}(x_n - \bar{x}_n), \ j = 1, 2, \ldots, n$$

Figure 1. Specification of a trained stretch and hammer neural network using Gaussian radial basis functions.

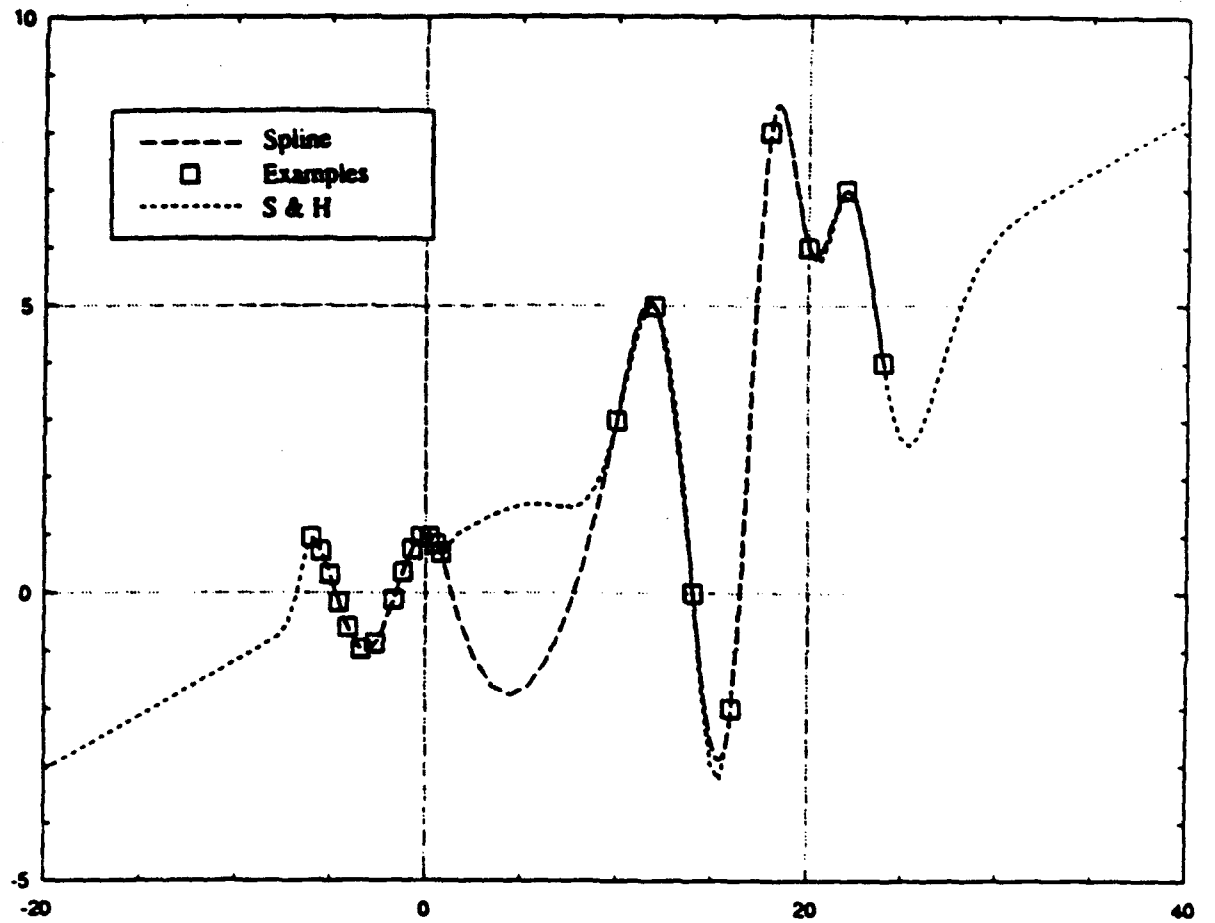


Figure 2. Comparison of stretch and hammer and natural cubic spline interpolation for one-input training examples.

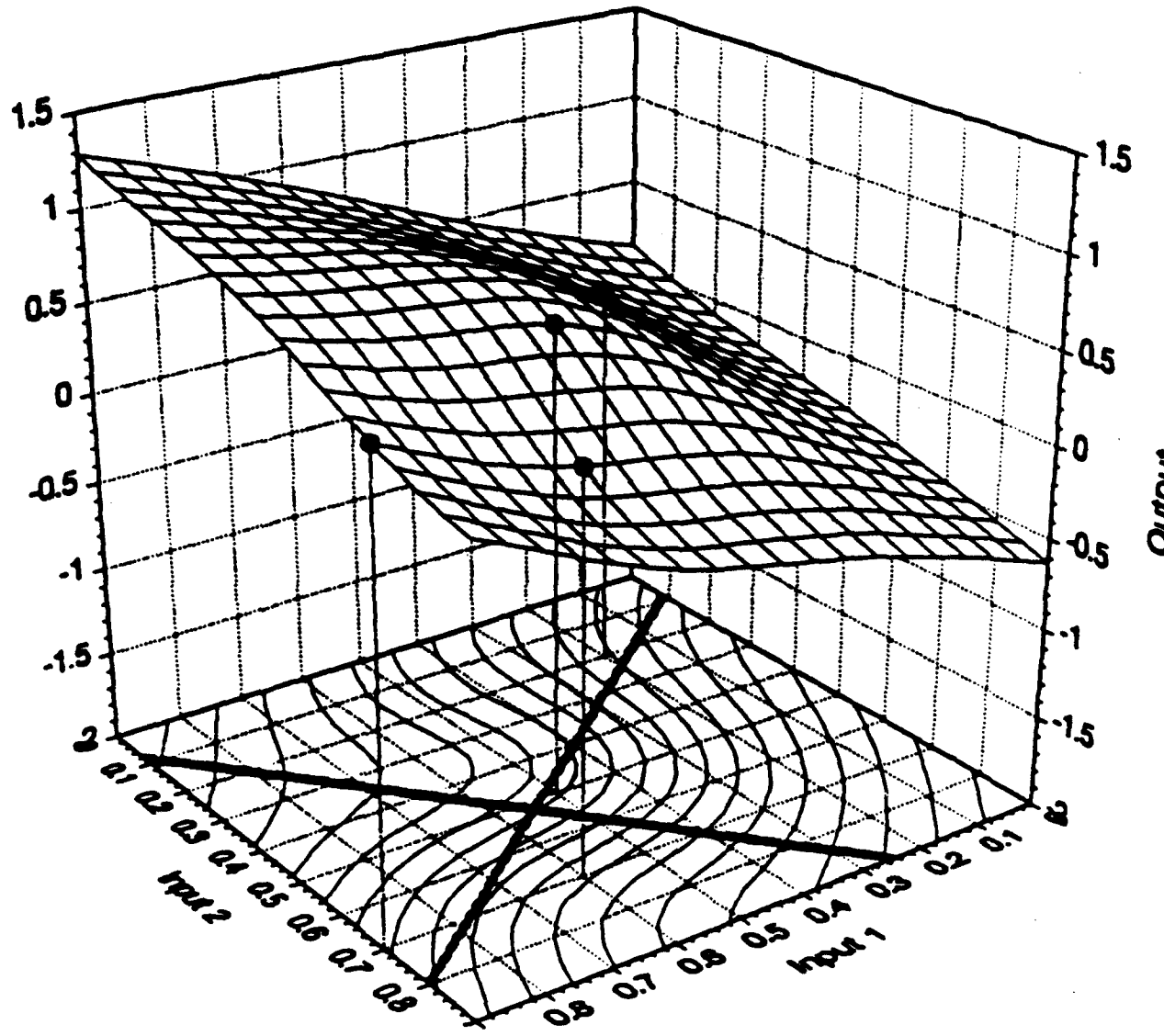


Figure 3. A least squares plane fitted and hammered to four two-input training examples with the two stretched (principal component) coordinates indicated.

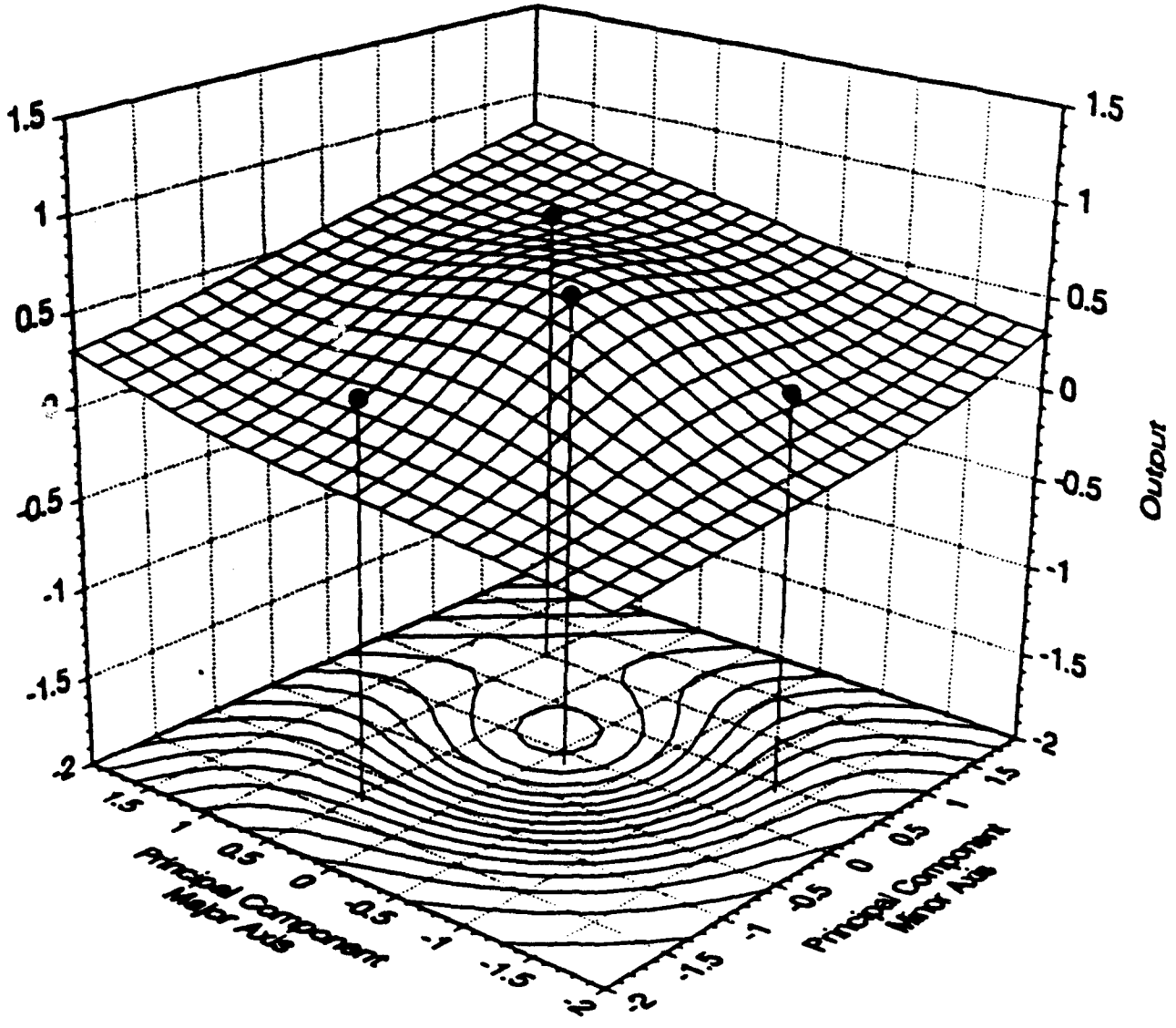


Figure 4. The training examples of Figure 3 in the stretched coordinates with a least squares plane fitted and hammered in these coordinates.

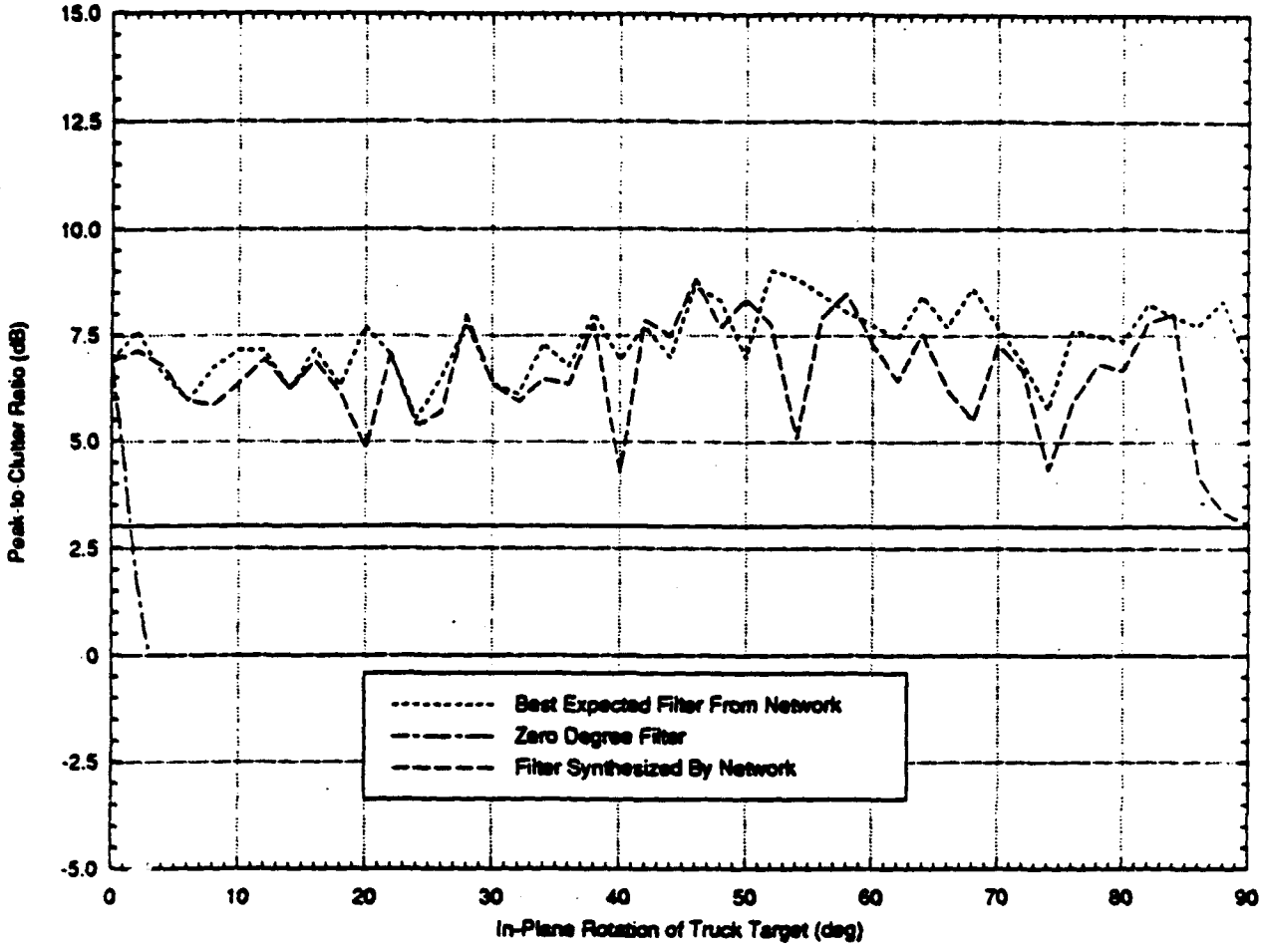


Figure 5. Correlation peak to clutter ratio versus in-plane target rotation angle for three correlation filters, where a clutter background not used in training was employed to test the filter synthesized by the stretch and hammer neural network.

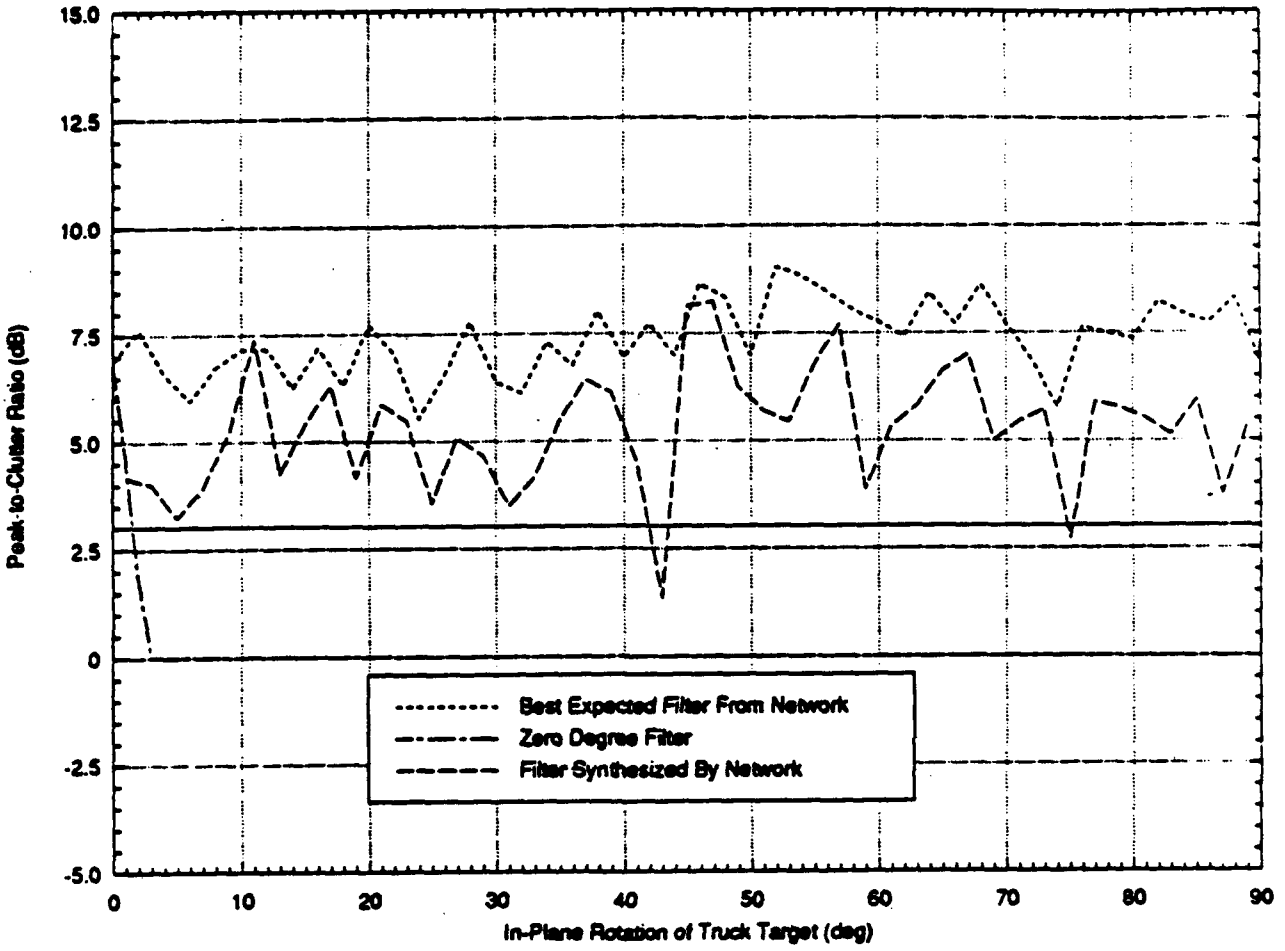


Figure 6. As in Figure 5 except that both a clutter background and target rotation angles not used in training were employed for testing.

OPTICAL FILTER SYNTHESIS USING ARTIFICIAL NEURAL NETWORKS

Thesis

Submitted to

Graduate Engineering & Research
School of Engineering

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree

Master of Science in Electro-optics

by

Mark August Manzardo

UNIVERSITY OF DAYTON

Dayton, Ohio

April 1992

OPTICAL FILTER SYNTHESIS USING ARTIFICIAL NEURAL NETWORKS

APPROVED BY:

Steven C. Gustafson, Ph. D.
Advisory Committe, Chairman

Franklin E. Eastep, Ph. D.
Associate Dean/Director
Graduate Engineering & Research
School of Engineering

Patrick J. Sweeney, Ph. D.
Dean
School of Engineering

ABSTRACT

OPTICAL FILTER SYNTHESIS USING ARTIFCIAL NEURAL NETWORKS

Manzardo, Mark, August
University of Dayton, 1992

The feasibility of using neural networks to synthesize filters for a hybrid adaptive correlator (HAC) was demonstrated. Input scene binarization, filter generation, filter encoding, input scene sampling, and target rotation were considered in developing neural networks for target tracking. Neural networks were trained using target-plus-background image samples as inputs and coded filter values as outputs After learning, input image samples not included in training were used to test the neural networks. The resulting coded filter values were evaluated using computer-simulated optical correlation with ternary phase amplitude filters (TPAF). Back-propagation and stretch and hammer neural networks successfully synthesized filters for optical correlation, and performance was adequate for tracking rotated targets on various backgrounds. Typical correlation peak-to-clutter ratios were 3 to 9 dB for in-plane target rotation angles of 0 to 90 degrees.

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Hybrid adaptive correlator technology

Pattern recognition by optical correlation is accomplished by intentionally modifying the spatial frequency spectrum of an image, and thus it is a subset of Fourier optical signal processing. The VanderLugt correlator introduced modern optical signal processing concepts [VanderLugt, 1963]. The correlation theorem in Fourier analysis states that the correlation of functions f_1 and f_2 is

$$C(x,y) = F^{-1}\{F[f_1(x,y)]\ F^*[f_2(x,y)]\}$$

where F- is the Fourier transform operator
*- represents the complex conjugate
F^{-1}- represents the inverse Fourier transform operator.

By using the properties of lenses and coherent light, the correlation function can be produced at the Fourier transform plane of the second lens illustrated in Figure 1.1. For pattern recognition f_1 is an input image and F_2 is the conjugate Fourier transform of the target being searched for in the input scene. In general F_2, the correlator filter, is complex valued. The process of using the actual amplitude and conjugate phase of a target as described above is called matched filter or VanderLugt correlation. Using a matched filter requires holographic recording [Goodman, 1968] which is not practical for real-time pattern recognition. However, magneto-optic spatial light modulators

(MOSLMs) can be used for real-time pattern recognition [Ross, 1983]. These devices are capable of modulating incident light using three states [Kast, 1989], e.g., full amplitude with 0 degree phase shift, full amplitude with 180 degree phase shift, and zero amplitude.

Recent work has shown that in the area of image analysis phase information is more important than amplitude information [Oppenheim, 1981]. Computer correlation simulations using only phase information led to the phase-only filter (POF) concept [Horner, 1984]. A POF is advantageous because it does not attenuate the amplitude of the optical beam. Phase only filtering can produce correlation peaks many times more intense than simple matched-filtering. Another typical advantage of phase only filtering is localization of the correlation peak. A POF often yields a sharp peak whereas a matched filter yields a wide peak [Flannery, 1989].

Modern real-time SLM devices do not allow the implementation of complete phase modulation. A subset of the POF, namely the binary phase-only filter (BPOF), can be used with modern devices [Flannery, 1988]. A BPOF filter requires only 1 bit for filter storage per pixel, whereas a continuous POF requires 4 or more bits. BPOFs are designed using a threshold line angle (TLA) parameter [Flannery, 1988] described in Section 4.

A matched filter by definition performs best using signal-to-noise ratio as a metric for the case of additive Gausian noise. However, POFs and BPOFs have performed better than matched filters when used with real-world backgrounds [Fielding, 1990]. The use of BPOF techniques is motivated by

possible immediate implementation using available electronically-addressed spatial light modulators (SLMs) in the hybrid adaptive correlator (HAC) system. BPOFs for optical correlation have been successfully implemented [Psaltis, 1984]. The HAC system uses electronics and optics to preform pattern recognition and is illustrated in Figure 2.1. The operation of this system is further discussed in Section 2.

The scene must be binarized to be implemented in a HAC system using MOSLMs. Binarization is accomplished using algorithms which edge-enhance and then binarize the output by thresholding pixel values. Various binarized scenes can be created by varying the threshold, including scenes that correlate well with a particular filter. However, correlation performance is usually compromised for other backgrounds. A discussion of techniques for binarizing input scenes is presented in Section 3.

The filters considered in this research are 128 by 128 ternary phase amplitude filters (TPAFs) consisting of a BPOF multilplied by a bandpass binary amplitude mask with a low spatial-frequency block radius of 10 pixels and a high-spatial-frequency cut-off radius of 60 pixels, and they are encoded to allow for neural network implementation. The BPOF filter part of a TPAF is created by thresholding the real part of the Fourier transform of the target, which implies a TLA of 0 degrees and thus a symmetric filter (so that storing or encoding only one-half of the filter values is necessary). For a 128 by 128 image this thresholding strategy implies a need to store 5400 separate filter values, since values outside the 10-60 pixel radius bandpass are set to zero.

Neural networks typically require excessive training time to learn this number of values. A discussion of this concern is given in Section 4.

1.2. Neural networks for HAC filter synthesis

A large number of filters are needed to accommodate rotations, scale changes, and other distortions in the input scene target. Instead of correlating with each of these filters, a neural network can be used to synthesize the filter for correlation as the input scene target distortions evolve. A consequent reduction in processing time could enable feasibility for the HAC system for real-time target recognition.

Originally, work conducted using neural networks for filter synthesis used network inputs from the correlation peak. On low intensity backgrounds this technique was successful, but for significantly cluttered backgrounds neural network performance deteriorated, even for strong correlation peaks [Olczak, 1991]. However, the research reported here indicates that the use of gray-level input plane intensity values centered on the location of the correlation peak enables acceptable neural network filter synthesis. Studies on the use of input plane samples for neural network inputs have been made [Olczak, 1991]. For this technique to be successful, an appropriate region on the input scene must be sampled. Using the location indicated by the correlation peak assumes that a target has already been recognized, and hence neural network filter synthesis is intended only for target tracking. However, if the correlation peak or some other technique indicates the location of a blob-

like object, then a neural network can be used to confirm that this object is or is not a target.

The input scene must be sampled to produce the neural network inputs. A variety of input scene sampling techniques were investigated. The simplest consisted of 25 pixel values on a 5 x 5 grid centered on the target. To incorporate more of the background in the input samples, a 9 x 9 grid was used, and two separate algorithms were employed to produce 25 or 31 input values for the neural network. The choice of a sampling technique is important for adequate neural network interpolation between samples. Sampling techniques are discussed further in Section 5.

Two different types of neural networks were used as interpolators to synthesize filters: the well known back-propagation neural network and a recently developed "stretch and hammer" neural network. The general idea was to input enough representative examples to train the neural networks to approximate a desired input/ output behavior for filter synthesis. The inputs are gray-level pixel values from the input scene to be binarized and coded for the input plane SLM. The outputs are the coded values that characterize the TPAF for the filter plane SLM. There are advantages to using the stretch and hammer as well as the back-propagation neural network. These advantages and an in-depth analysis of neural network design is presented in Section 6.

Successful neural network filter synthesis was accomplished using both back-propagation and stretch and hammer neural networks. A variety of input/ output relationships were established. Testing of neural network performance

was accomplished by varying input scene parameters that included in-plane target rotation angle and target background type. The results are presented and tabulated in Section 7.

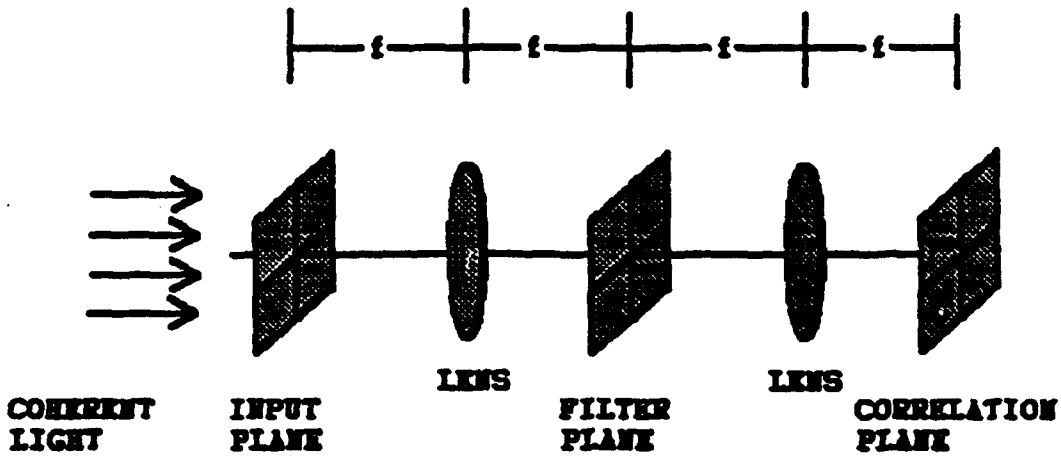


Figure 1.1 Optical set-up for correlation.

2. HAC SYSTEM OPERATION

The HAC is an electro-optical pattern recognition system that integrates electronics and the computational power of lenses. From Fourier Optics it is known that using a coherent light beam, a lens can transform an input 2-D pattern into its Fourier spatial spectral components at the focal plane of the lens. Each unique input object (target) has associated with it a unique Fourier spectrum. By filtering the spectrum for a certain object in the image, it is possible to recognize the presence of the object. This process is known as pattern recognition by optical correlation, and can be implemented using the optical set-up in Figure 2.1a.

2.1. Steps in HAC operation

The first step in the operation of the HAC system is the acquisition of the input scenes. The images used in this research are gray-level visible and infrared images originating from two sources: the University of Southern California Image Processing Institute Data Base (USC), and the United States Army Center for Night Vision & Electro-Optics 1987 Multi-Sensor Field Test (ARMY) (see Appendix for examples). These images comprise a "test-world" for research purposes.

The second step in the operation of the HAC system is processing the available input scenes. A computer is used to binarize the images and to download the results to the input plane SLM. The binarization process is discussed in Section 3. The binarized image is stored for later use in the process of optical correlation by computer simulation.

The third step in the operation of the HAC system involves hybrid electro-optical functions. As shown in Figure 2.1a, a HeNe laser beam (wavelength 632.8nm) is expanded and collimated to illuminate the input scene SLM. The limited modulation capability of the SLM requires a binary image. This SLM acts as a transparency so that the output is a pure amplitude function encoding the binarized image. The first lens generates the Fourier transform of this function at the filter plane SLM (in a time that equals the input-filter plane distance divided by the speed of light). The computer processor retrieves a filter from a previously stored bank of filters. The filters are ternary phase amplitude filters (TPAFs) and are further discussed in Section 4. These filters modulate the phase using two phase states or block the light completely. The second SLM implements a filter by simple multiplication with the Fourier transform of the input scene. If only the target is present the output of the filter SLM approximates a plane wave with a direction related to the target position. The second lens produces the Fourier transform of the output of the second SLM at the 2-D detector. If a target is present in the input scene, then a plane wave is the output of the second SLM and a delta function or bright spot appears on the detector at the target location.

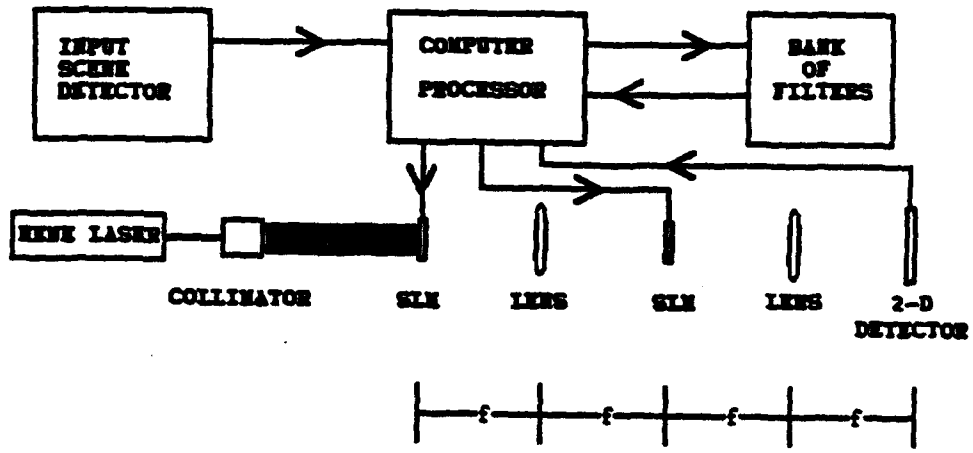
Step four of HAC system operation requires a decision process. The output of the 2-D detector is processed using a peak-finding routine. If a peak is found that is distinguishable from clutter, then target recognition is accomplished; if no peak is found then the processor must download the next filter from the bank of filters to the filter plane SLM. The process of steps 3 & 4 is carried out until either a target is recognized or all of the filters are scanned with no recognition accomplished. There exist smart filtering strategies which do not require all filters to be scanned.

2.2. Neural networks in HAC operation

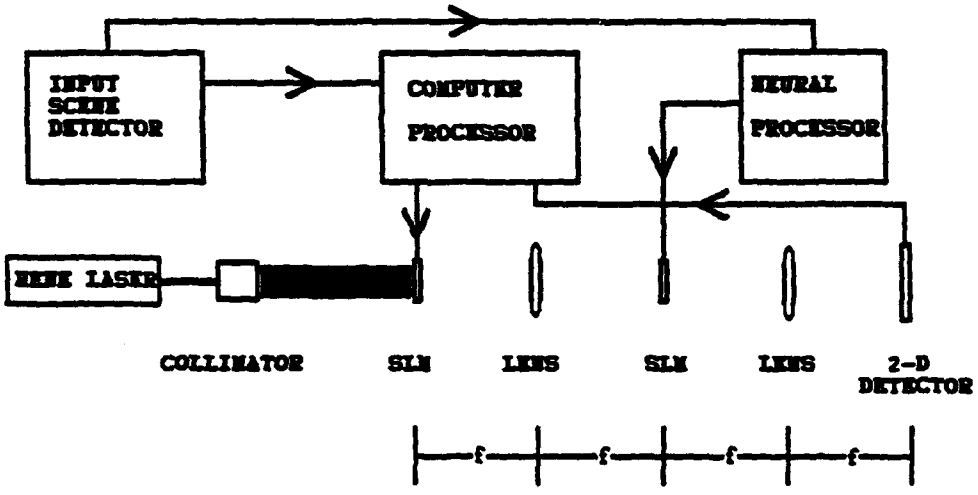
The process in step 3 can be simulated by computer using appropriate digital image processing algorithms. The Fast Fourier Transform (FFT) algorithm accomplishes the task of the first lens. This algorithm inputs the stored real-valued binarized scene (created in step 2 of the HAC process) and outputs the complex-valued Fourier transform of the scene. This complex-valued function is then multiplied pixel-by-pixel with one of the filters from the filter bank, and the Inverse Fast Fourier Transform (FFTI) is then performed. The squared modulus operation is performed on the FFTI output, which simulates the 2-D detector recording (assuming that the 2-D detector responds linearly with irradiance). This output is then processed as in step 4 of the HAC system. Once again the steps 3 & 4 are repeated until a target is recognized or the bank of filters is exhausted.

Typically, extensive processing time is needed not only to scan through the filters but also to search each output of the 2-D detector array for a

correlation peak. Figure 2.1b illustrates the concept of incorporating a neural network processor in the HAC system to reduce the number of correlations performed. The research performed here assumes that the location of a potential target is known. The input scene is gray-level and at the disposal of the neural processor, and following a consistent sampling technique using these gray-level values a neural network can be used to synthesize a filter for correlation in a short time. Only one filter is synthesized by the network, and hence only one complete correlation is necessary to determine whether or not a target is in the scene. Selection of the sampling technique is important and is discussed in Section 5. Selection of the neural network also affects the results, and a discussion of appropriate neural networks is given in Section 6.



(a)



(b)

Figure 2.1 Hybrid adaptive correlator (a) with a filter bank, (b) with a neural processor.

3. INPUT SCENE BINARIZATION

3.1. Binarization techniques

Two important image storage techniques are used in digital image processing. The first method stores an image by coded variations in intensity, which is accomplished by assigning one of many gray-level intensity values to each pixel in the image. This type of coding is limited by the dynamic range of the acquisition equipment or in some cases by the storage system. The pixels in the images used in this research are assigned gray-level pixel values from 0 (black) to 255 (white). Each pixel requires one byte of storage space. For an image of 128 by 128 pixels the required storage space is 16,384 bytes (not including 124 bytes for the image file header). Many optical devices, such as the MOSLMs, cannot represent a gray level image and hence a more restrictive coding method must be employed. The second coding method thresholds each pixel at some value between 0 and 255 so that the resulting image file is filled with either 0's (representing "off") or 255's (representing "on") at each pixel location.

Gray-level images contain more target information and it would be advantageous to keep the images in gray-level format for use in the optical correlator. If this were possible the binarization prepro g step would not be necessary. The most important reason for binarizing images is due to

restrictions on the SLMs used in the input and filter plane of the HAC system. At each pixel location these devices either block the light completely, let the light pass through unchanged, or phase shift the light by 180 degrees, and it is not possible to download a gray-level image to such a device. The input plane SLM creates a pure binary amplitude encoded image similar to the coding that could be obtained by placing a mask with transparent and opaque regions at the SLM location.

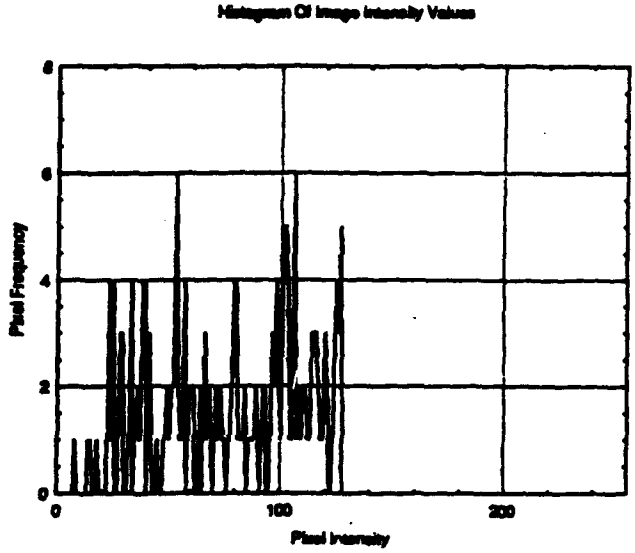
Studies on binarization techniques have been made [Johnson, 1991]. A simple technique uses the original gray-level input image values and thresholds on a value between 0 and 255. The binarized image has different characteristics for different threshold values. A low threshold value tends to transform a large number of the pixels to the 255 or "on" value and yields "blob-like" binarized image characteristics. A high threshold value tends to result in a binarized image with less information and, in some instances, too little information to recognize a t[illegible] Thus, a simple thresholding value for binarization is not adequate for implementation in the HAC system. Histograms of pixel values for typical backgrounds and targets in Figures 3.1 through 3.5 show that the background and target pixel intensity distributions commonly overlap. Thresholding at a particular value tends to emphasize the background information as much as the target information. Thus another approach which separates the pixel intensity distributions of the background and target before binarization is needed.

3.2. Edge-enhanced-and-threshold binarization technique

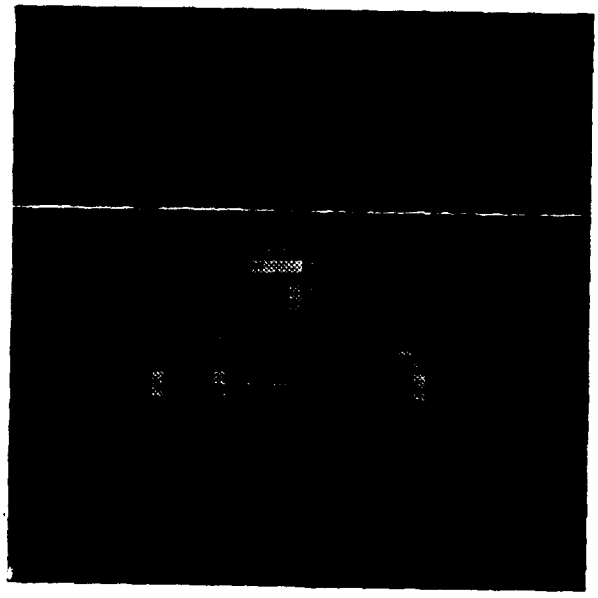
A successful binarization technique has been developed [Johnson, 1991]. In this technique the input image is edge enhanced before thresholding, as described below. Edge enhancement is accomplished using a 2 x 2 square grid of pixel values from the input image. The maximum difference of pixel intensity within the grid is computed. This value replaces the upper left corner value of the grid in the new edge-enhanced image. Histograms of pixel values of backgrounds after edge enhancement illustrated in Figures 3.1 through 3.5 show a strong shift toward a low intensity distribution, which is helpful in selecting a threshold level. Using statistics a threshold level can be chosen such that a large portion of the background pixels can be set to zero while not severely degrading the target. This level is selected using the expression

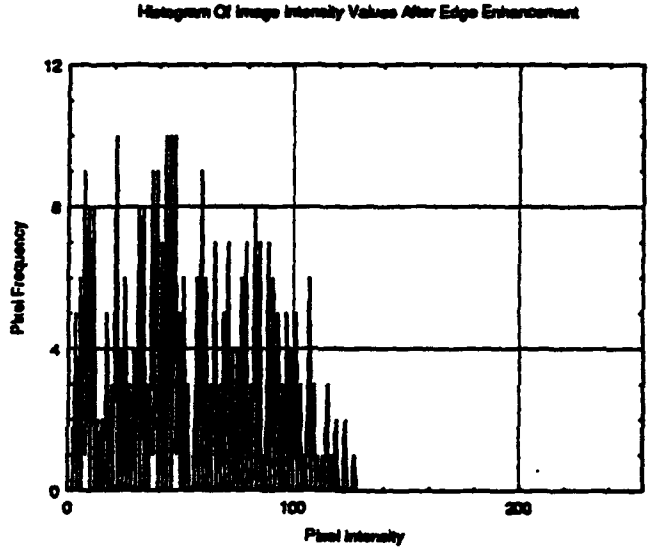
$$\textit{Threshold} = \textit{mean} + \textit{SDM} * \textit{standard deviation},$$

where SDM is defined as the standard deviation multiplier. Appropriate selection of the SDM yields improved results for optical correlation using the peak-to-clutter ratio as a metric. An SDM of 1.8 as illustrated in Figure 3.6 was found to yield the best results using the peak-to-clutter ratio metric and was used for the research reported here. The peak-to-clutter ratio is the ratio in decibels of the energy in the target peak to the energy in the largest clutter peak.



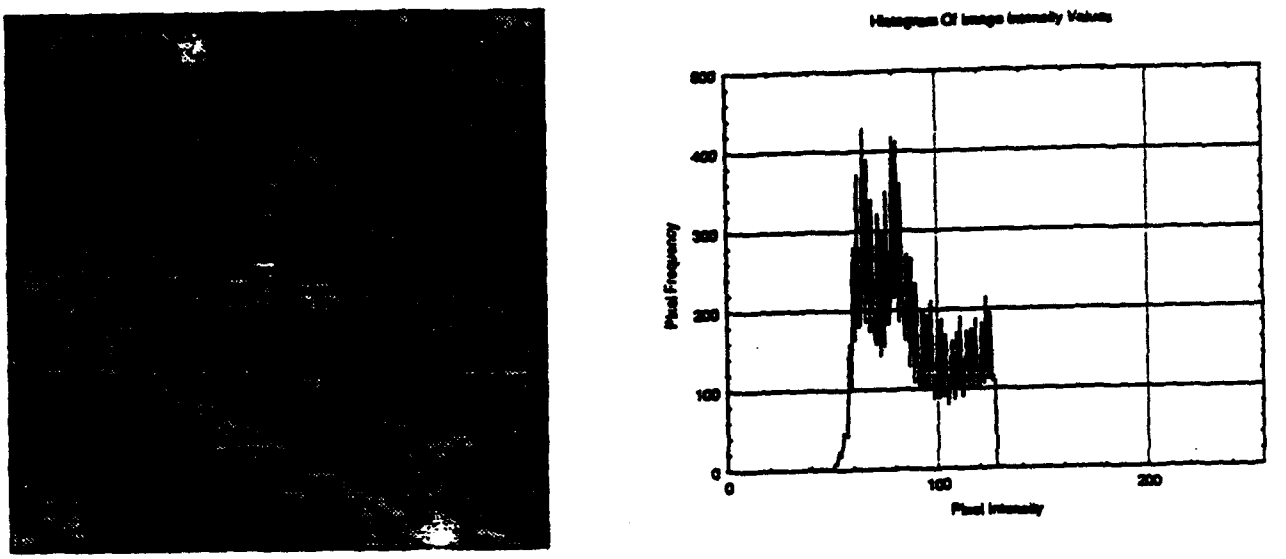
(a)



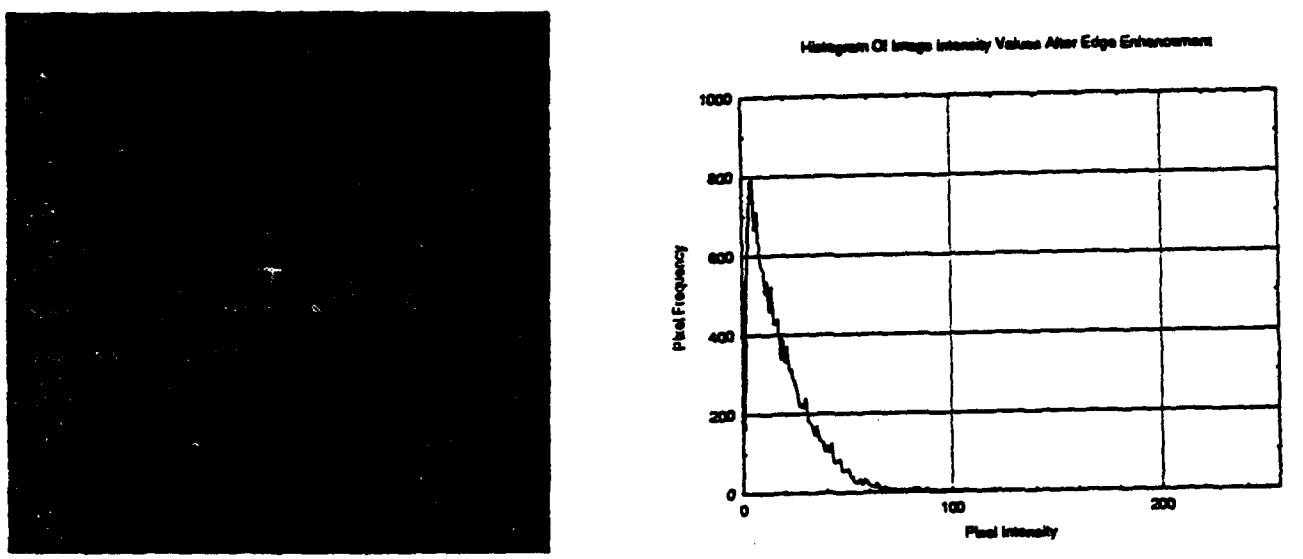


(b)

Figure 3.1 (a) Truck target with histogram, (b) truck target after edge enhancement with histogram.

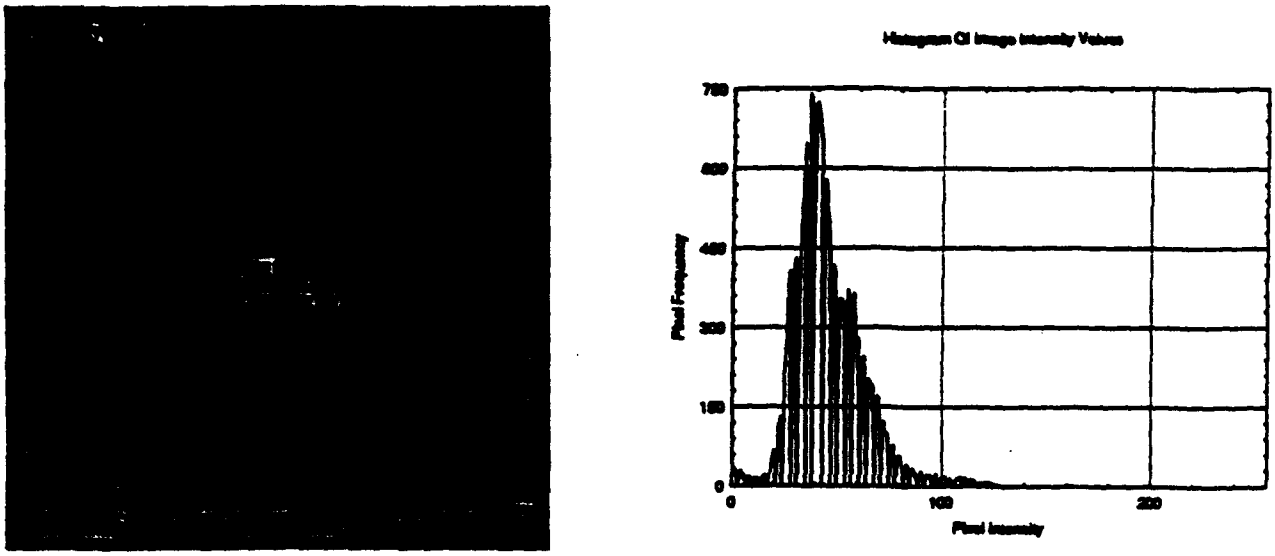


(a)

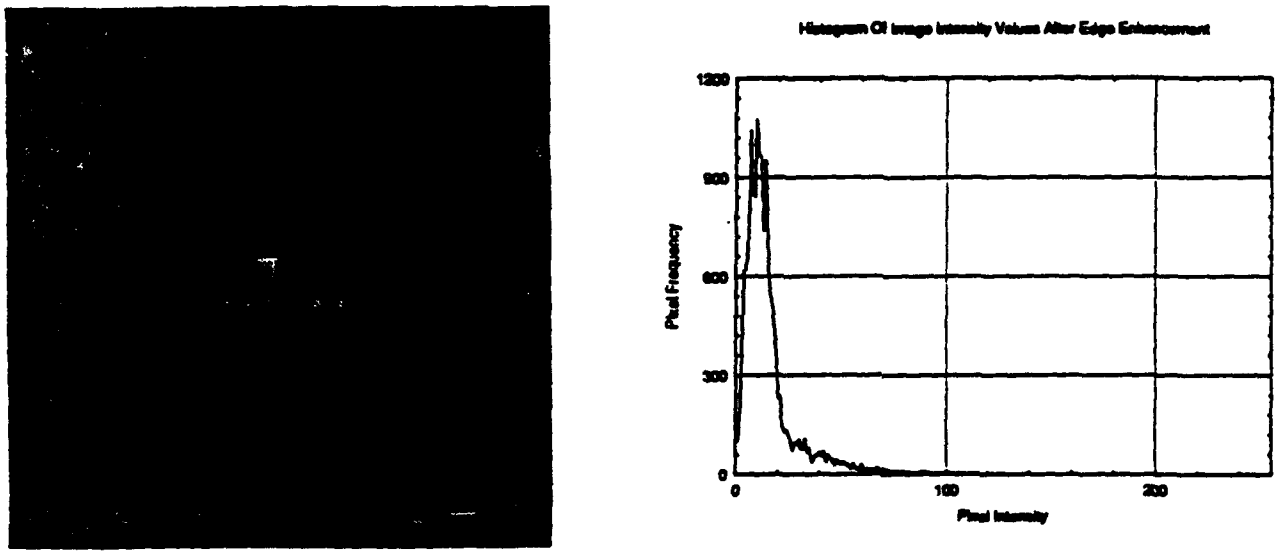


(b)

Figure 3.2 (a) Truck on city70 background with histogram, (b) truck on city70 background after edge enhancement with histogram.

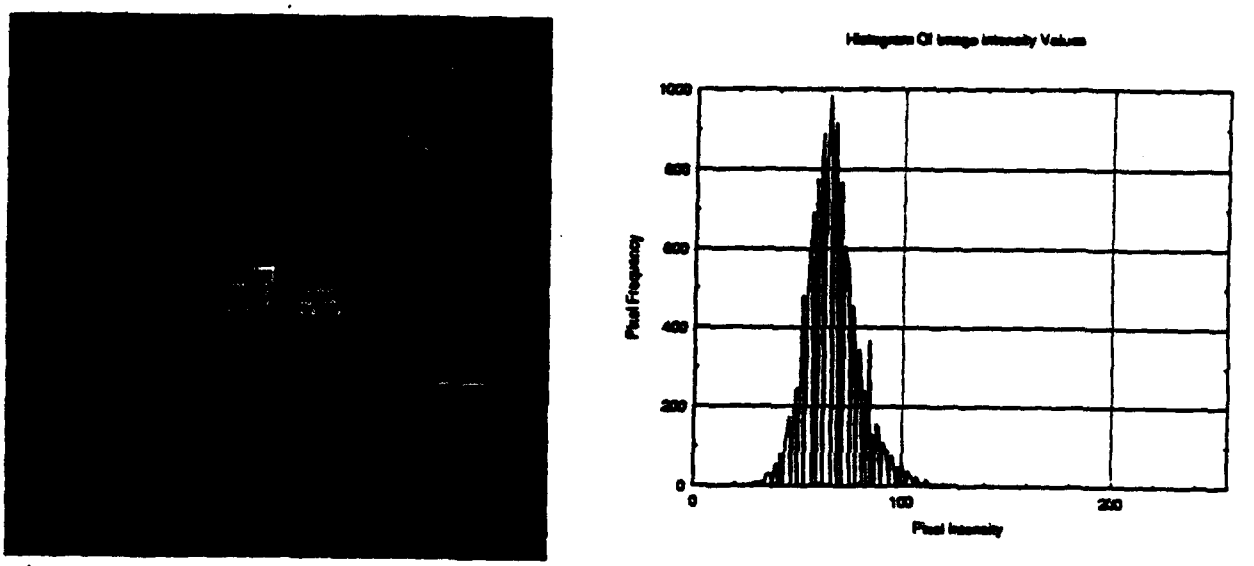


(a)

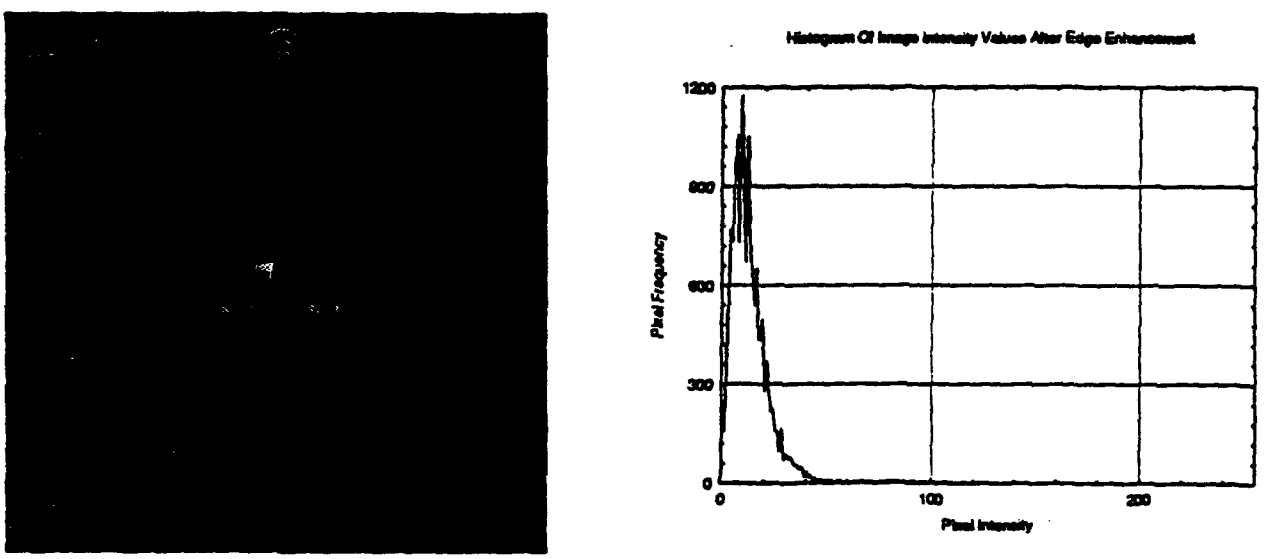


(b)

Figure 3.3 (a) Truck on trbk70 background with histogram, (b) truck on trbk70 background after edge enhancement with histogram.

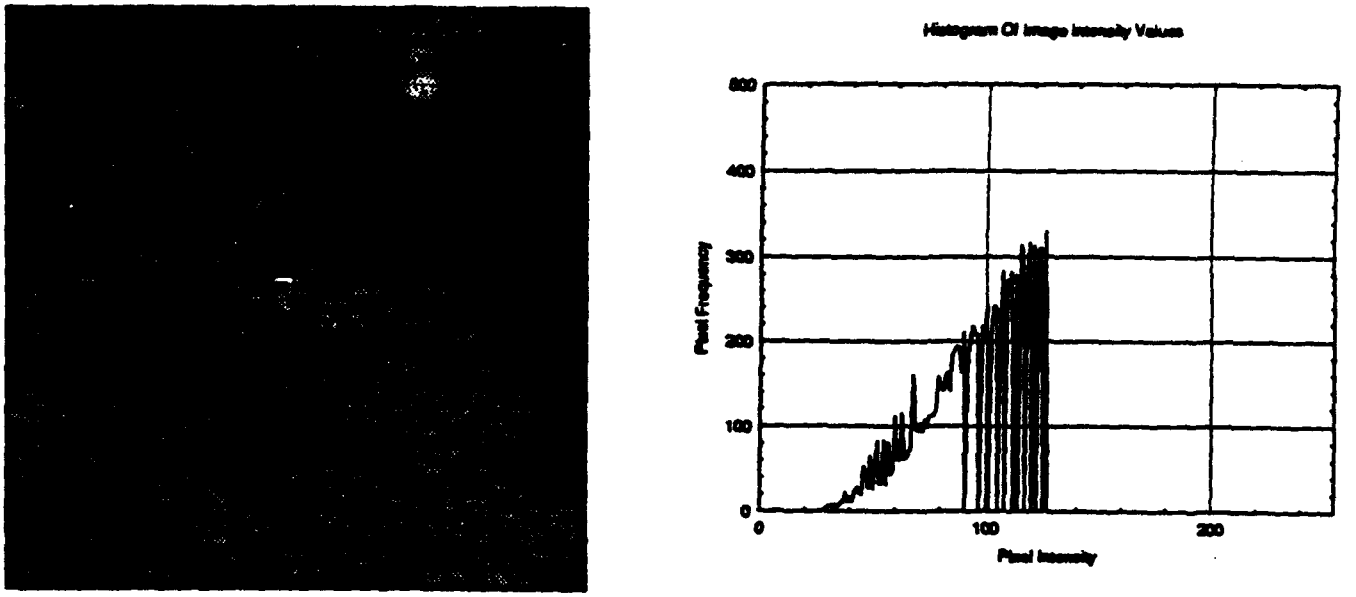


(a)

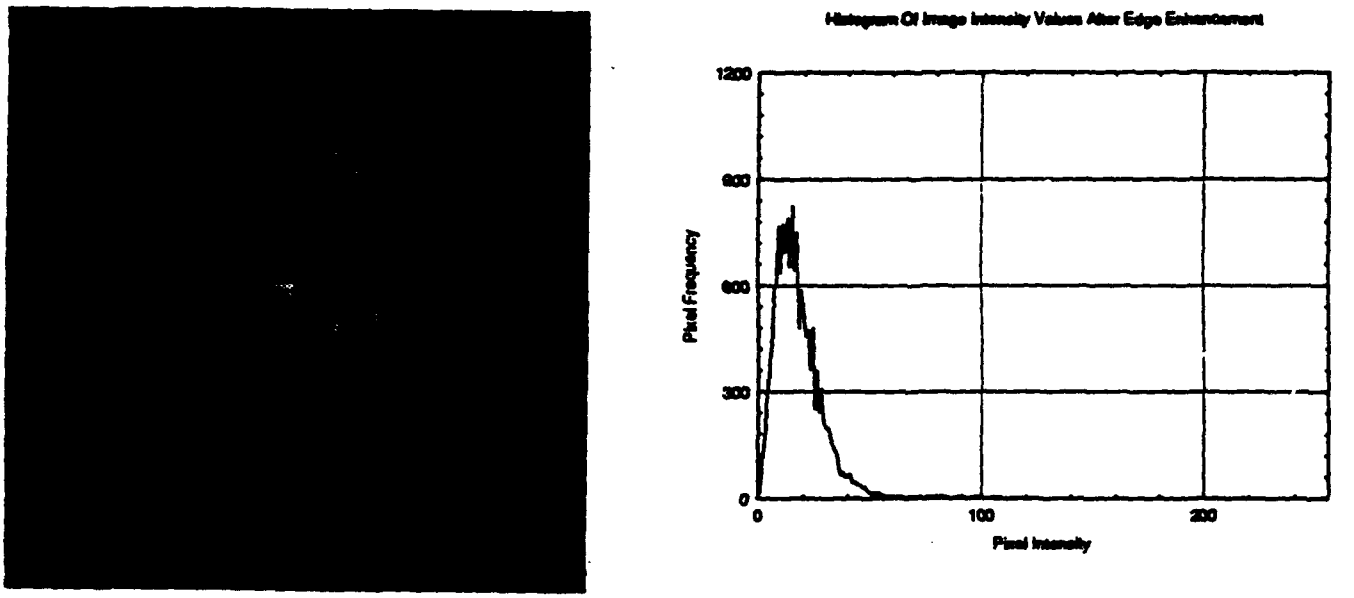


(b)

Figure 3.4 (a) Truck on newbk80 background with histogram, (b) truck on newbk80 background after edge enhancement with histogram.



(a)



(b)

Figure 3.5 (a) Truck on bushy background with histogram, (b) truck on bushy background after edge enhancement with histogram.

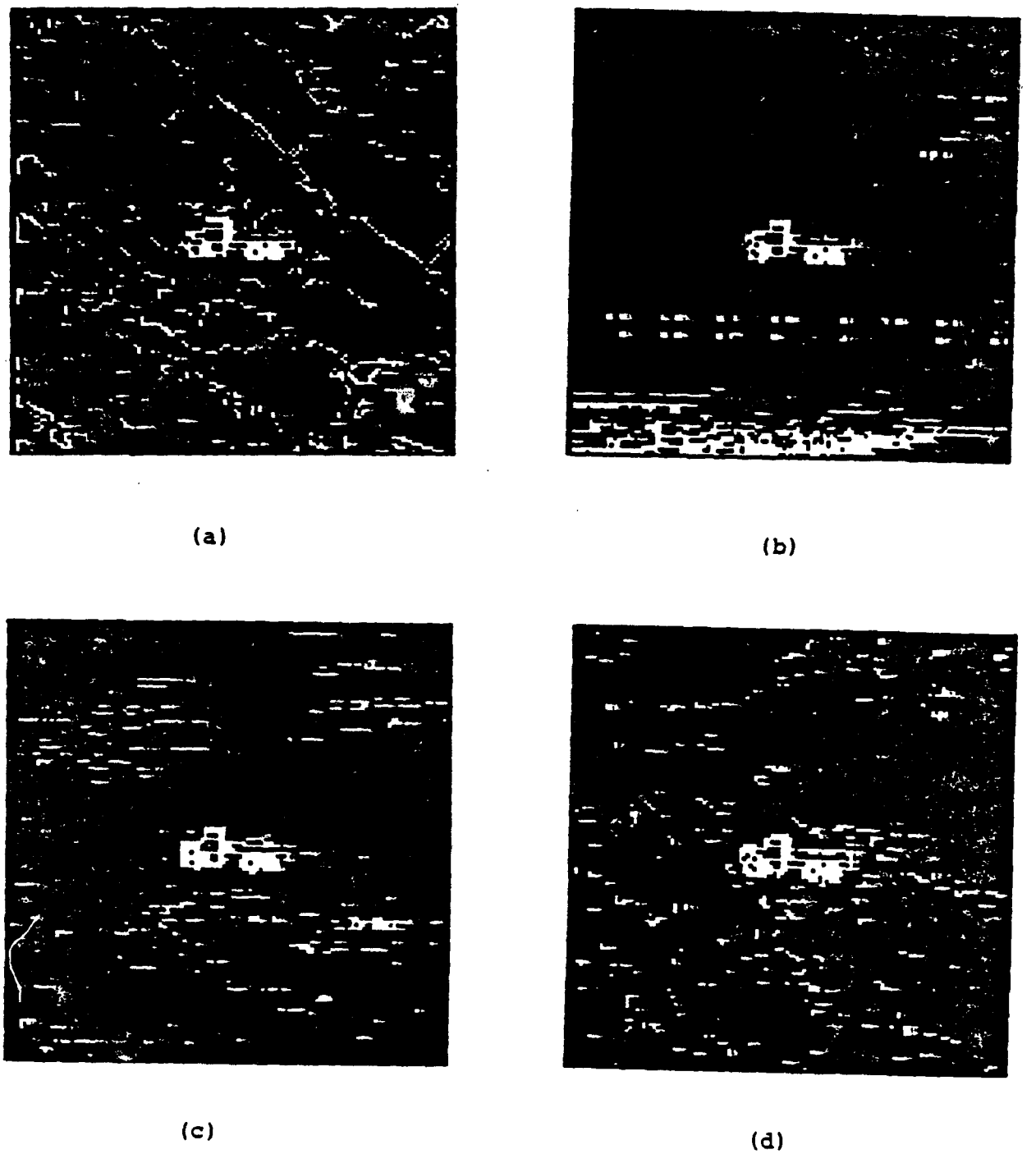


Figure 3.6 Binarized tucks using an SDM of 1.8: (a) on city70 background, (b) on trbk70 background, (c) on newbk80 background, (d) on bushy background.

4. FILTER ENCODING

4.1. Encoding requirements and techniques

The HAC system uses a MOSLM at the correlator filter plane. Each pixel in the SLM may be operated in three states: two states that rotate the plane of polarization approximately + or - 6 degrees and one state that scatters the light out of the system [Ross, 1983; Psaltis, 1984; Kast, 1989]. Light incident on the filter plane represents the Fourier transform of the binarized input image. Light passing through the input plane SLM is linearly polarized, and light passing through the filter plane SLM is analyzed using an orthogonal polarizer. Each of the two SLM states that rotate the plane of polarization contains a component along the pass axis of the analyzer. There is a relative phase shift of 180 degrees between the two states, and thus a relative amplitude change is produced. In effect, the pixels set to one state pass the Fourier transform without alteration, and the pixels set to the other state alter the Fourier transform by a phase shift of 180 degrees or amplitude multiplication of -1. The third state scatters most of the light out of the system and thus can be represented by zero. Therefore, using polarized light, an SLM, and an analyzer, it is possible to encode a filter with amplitudes of +1, -1, and 0 at each pixel location. The TPAF is defined by these values and thus consists of a BPOF multiplied by a binary amplitude pattern.

A TPAF filter for pattern recognition is generated by first taking the Fourier transform of a target, which is accomplished using the FFT algorithm. In general, the target has a complex-valued Fourier transform so that each pixel value is represented by a point in the complex plane (e.g. a 128 by 128 image has 16,384 complex valued samples). Producing a BPOF from these values requires choosing a line through the complex plane so that all pixel values on one side of the line are assigned a phase of 0 degrees (amplitude unchanged) and all values on the other side are assigned a phase of 180 degrees (amplitude multiplied by -1). The angle that this line makes with the imaginary axis is the Threshold Line Angle (TLA) as illustrated in Figure 4.1 [Flannery, 1988]. Varying the TLA makes it possible to achieve improved correlation performance [Flannery, 1989]. A TLA of 0 results in a cosine BPOF that is symmetric and hence reduces the storage requirements of the system by one-half, so that only 8192 values are needed to characterize the BPOF portion of the filter. A TPAF filter is then created by multiplying the BPOF filter by a 10-60 pixel radius bandpass amplitude pattern which sets those pixels outside this band to the zero state. This bandpass was chosen to reduce the number of necessary coded values needed to represent the filter.

4.2. Adopted encoding technique

The above procedure yields an excessive number of outputs for neural network implementation, and an encoding technique is needed to reduce this number. The bandpass or binary amplitude pattern for a 128 by 128 pixel filter has a low-spatial-frequency block radius of 10 pixels and a high-spatial-

frequency cut-off radius of 60 pixels. To further decrease the number of pixel values, a superpixel filter is produced by defining 3 x 3 superpixels and storing only one value of +1 or -1 per superpixel. The use of a superpixel filter blurs the impulse response of the filter, but the correlation performance is still acceptable. A superpixel filter allows for compression of the number of stored values from 8192 to 600, which is suitable for neural network implementation on desktop computers. When the filters are used for correlation each of the 600 superpixel values is expanded to form the nine pixels it represents. Figure 4.2 illustrates a typical 128 by 128 BPOF and the reduced 10-60 bandpass superpixel filter used for this research for a truck at 0 degree rotation.

The input scenes are binarized in accordance with the procedure described in Section 3. This procedure uses a threshold value related to the mean and standard deviation of the edge-enhanced input image, so that for different backgrounds the target is binarized using different thresholds. Variations in the target due to binarization must be accounted for when creating a filter. By superimposing the target on different backgrounds and examining the image after binarization, it is possible to create a binarized target for filter generation that correlates well with a variety of backgrounds. The approach taken here binarizes the target for filter generation using many SDM multipliers. Each of these binarized targets is used to generate a filter, and each of these filters is used in a computer simulated correlation with the target superimposed on a variety of backgrounds.

The SDM multipliers for filter generation varied from 3.0 to 5.0. Different SDM multipliers were needed for filter generation because in this process background information is not present. The binarization process was repeated for each orientation of the target used in training a neural network. The performance of the filter generated for a particular SDM may be good for a particular background but poor for another. By examining the peak-to-clutter ratio for each SDM and each background, it was possible to choose an SDM and generate a filter that had acceptable correlation performance for any of the backgrounds illustrated in the Appendix.

The choice of the SDM for filter generation was important for acceptable correlation performance and varied chaotically with rotation of the target. Thus it was necessary to fix the SDM value for the input image for use in a real system. The SDM value for the input image varied slightly from 1.5 to 2.1, but the best results using peak-to-clutter as a metric were at 1.8. Table 4.1 shows SDM values used for filter generation for target rotations from 0 to 90 degrees and for an input image binarization SDM of 1.8 that produced the best correlation performance. For 0 degree target rotation an SDM of 3.6 was used to produce the binarized target illustrated in Figure 4.3. This target can be compared to the binarized truck in the input scenes illustrated in Figure 3.6.

Angle	SDM	Angle	SDM	Angle	SDM	Angle	SDM
0	3.6	2	4.0	4	4.8	6	4.4
8	4.4	10	4.2	12	4.4	14	3.8
16	4.6	18	4.6	20	4.4	22	4.0
24	4.2	26	3.6	28	4.0	30	4.0
32	4.4	34	3.8	36	4.2	38	4.0
40	4.2	42	3.8	44	4.0	46	3.6
48	4.6	50	3.8	52	4.0	54	4.4
56	3.8	58	3.6	60	3.8	62	3.6
64	4.0	66	3.6	68	4.2	70	3.8
72	3.8	74	3.8	76	4.4	78	3.6
80	4.4	82	4.0	84	4.6	86	3.6
88	4.6	90	4.4				

Table 4.1 SDM values for filter generation using an input plane SDM of 1.8.

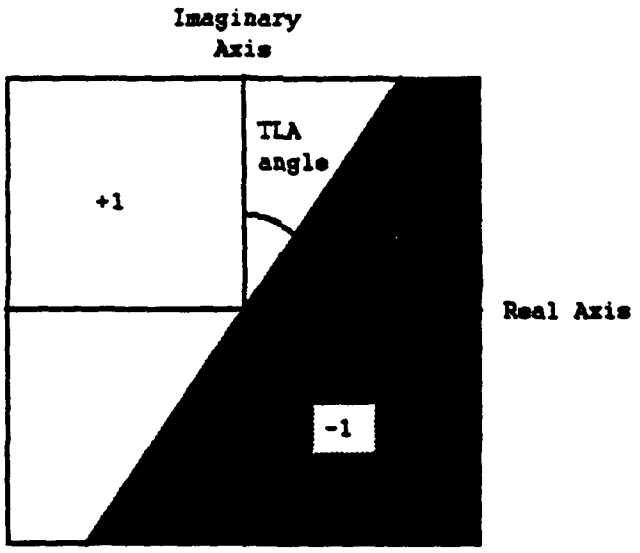
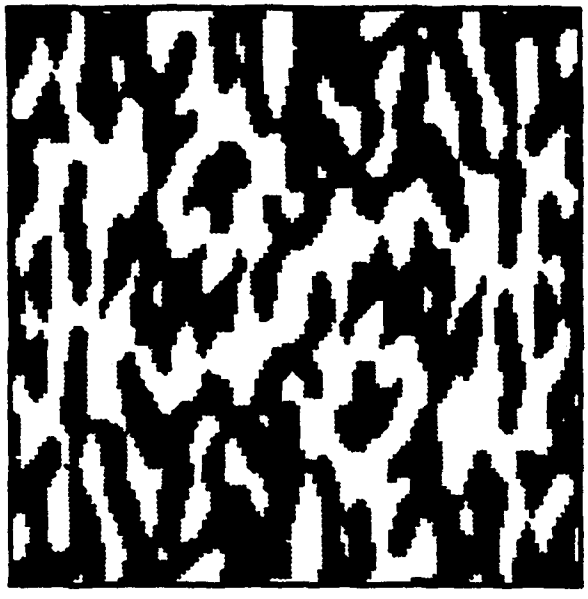
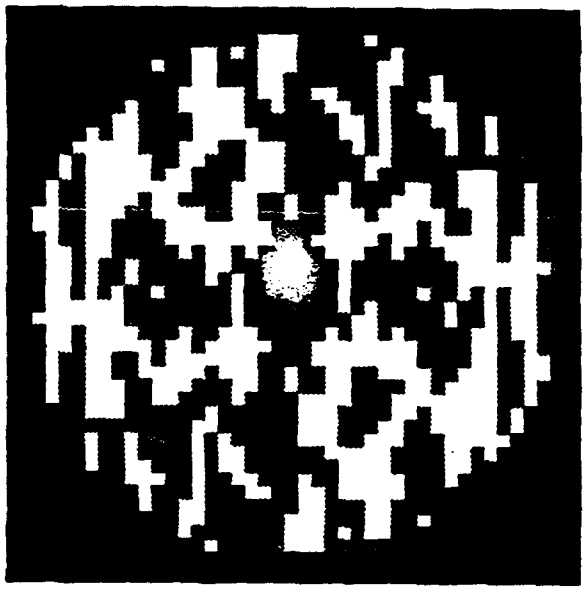


Figure 4.1 Illustration of TLA binarization of Fourier transform.



(a)



(b)

Figure 4.2 (a) BPOF made from the binarized Fourier transform of a truck at 0 degree rotation, (b) the reduced 10-60 bandpass TPAF made from (a).

Figure 4.3 Truck target binarized using a SDM of 3.6.

5. INPUT SCENE SAMPLING

5.1. Sampling requirements and techniques

To effectively train a neural network, suitable input and output variables must be identified. The goal in this research is to simulate the relationship between the input plane and the optimum filter for the optical correlator. This relationship may be established by determining the orientation of a target, creating an appropriate filter, and downloading the filter to the filter plane SLM. This process may be too slow to be used successfully in the HAC system. However, by choosing representative examples of input scenes it is possible to train an artificial neural network to learn the input/ output behavior, and this learning can be used in real time in the HAC system.

Neural network inputs for filter synthesis may be selected in many ways. All neural network inputs for this research were obtained from the input gray-level image. One sampling method, illustrated in Figure 5.1, used the gray-level pixel values in a 5 x 5 grid centered on the target. All except the center pixel value changed their values as the target was rotated, and on average the fraction of off-target pixel values was 10-15 percent. The 25 intensity values were from the box regions indicated on the images and in the illustration. For all backgrounds there was a discernable difference in each of the 25 intensity values for each two degrees of rotation. In both cases

illustrated, the upper right corner of the sampling grid extended into the background, and pixel values on the background were constant. Using part of the background for neural network inputs enabled the incorporation of noise. The 5 x 5 grid sampling technique allowed for successful filter synthesis using back-propagation neural networks if gray-level images were used in the input plane and TPAFs were used in the filter plane.

Unfortunately, gray-level input plane images can not be implemented using the SLMs described for the HAC system. A simple 5 x 5 pixel grid does not adequately represent input/ output behavior for neural network training for filter synthesis with the binary input images that must be used in the HAC system. Thus, for binary correlator input plane images it is necessary to use a different input plane sampling technique.

5.2. Adopted sampling technique for binary correlator input images

Two techniques were developed with the goal of training a neural network to perform well independent of background.

One technique used a 9 x 9 sampling grid centered on the target and an algorithm that reduced the 81 values in the grid to 25 values. This transformation of 81 to 25 values is illustrated in Figure 5.2. The transformation algorithm, which was designed to yield gradual but significant pixel value changes as the target was rotated, proceeds as follows. For each row of the 9 x 9 array the average deviation from the average row value is computed. The same computation is performed for each column, diagonal direction, and center 5 x 5 grid of values. The total number of values generated

is thus 25 (9 rows + 9 columns + 6 diagonal directions + center 5 x 5 grid = 25). Examples of actual backgrounds with the sampling grid superimposed are illustrated in Figure 5.3. It was found that this sampling technique allows the network to perform relatively independently of angle, but the technique degrades when tested with the target superimposed on different backgrounds not included in the training set.

A second sampling technique was developed to address the degradation problem. This technique sections the 9 x 9 grid into 36 wedges and computes the total intensity value in each wedge. The horizontal and vertical wedges depend only on one row or column and are not used for inputs to the neural network, thus avoiding the problem of the previous sampling technique where all inputs depended on only one row or one column. The 32-value input plane sampling technique is illustrated in Figure 5.4. It was found to be successful for neural network interpolation independent of target background or noise.

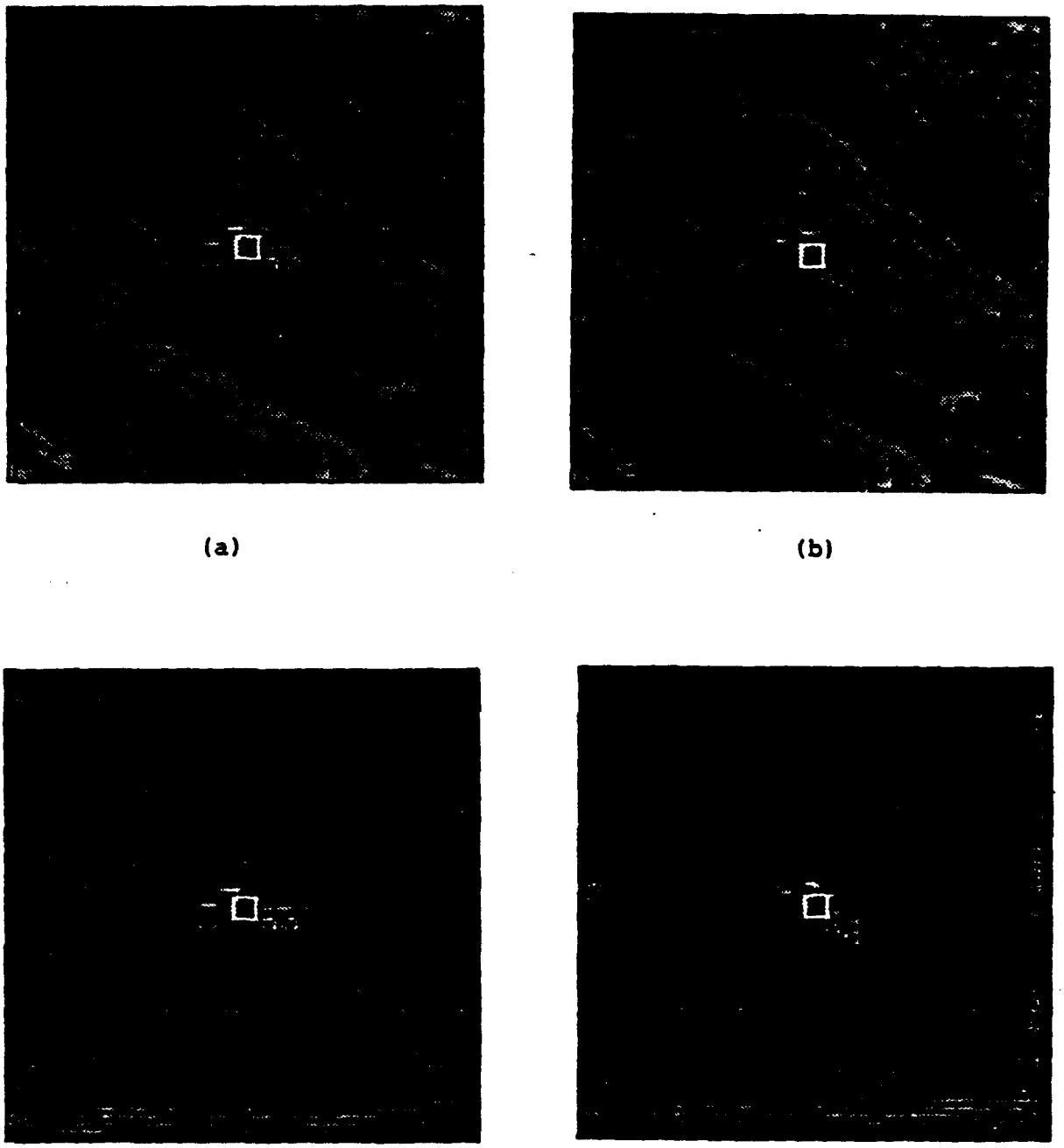


Figure 5.1 A 5 x 5 pixel grid superpixel on (a) truck on city70 background at a rotation of 0 degrees, (b) truck on city70 background at a rotation of 20 degrees, (c) truck on trbk70 background at a rotation of 0 degrees, and (d) truck on trbk70 background at a rotation of 20 degrees.

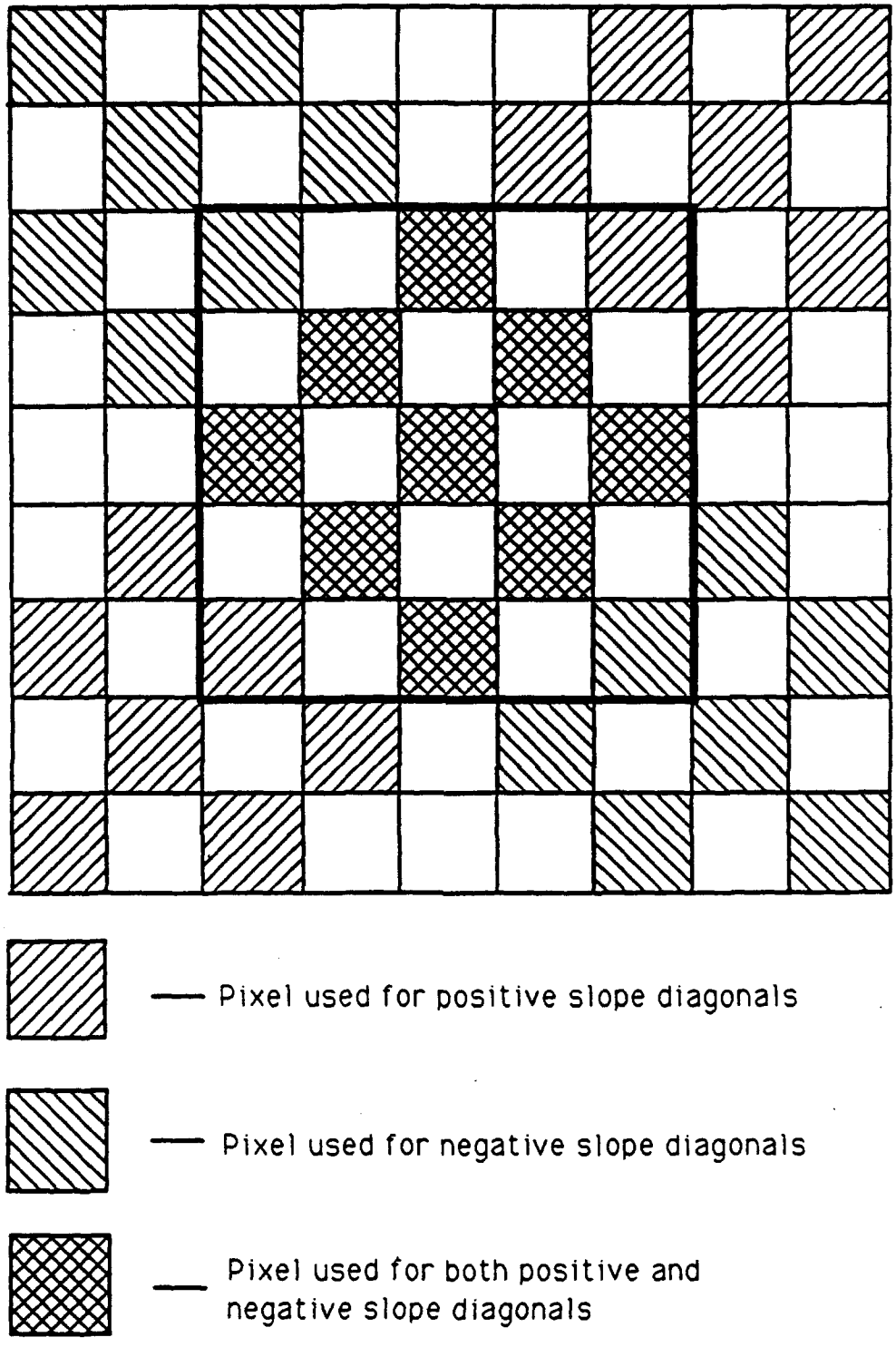


Figure 5.2 **9 x 9 pixel grid used to obtain 25 input values for neural network.**

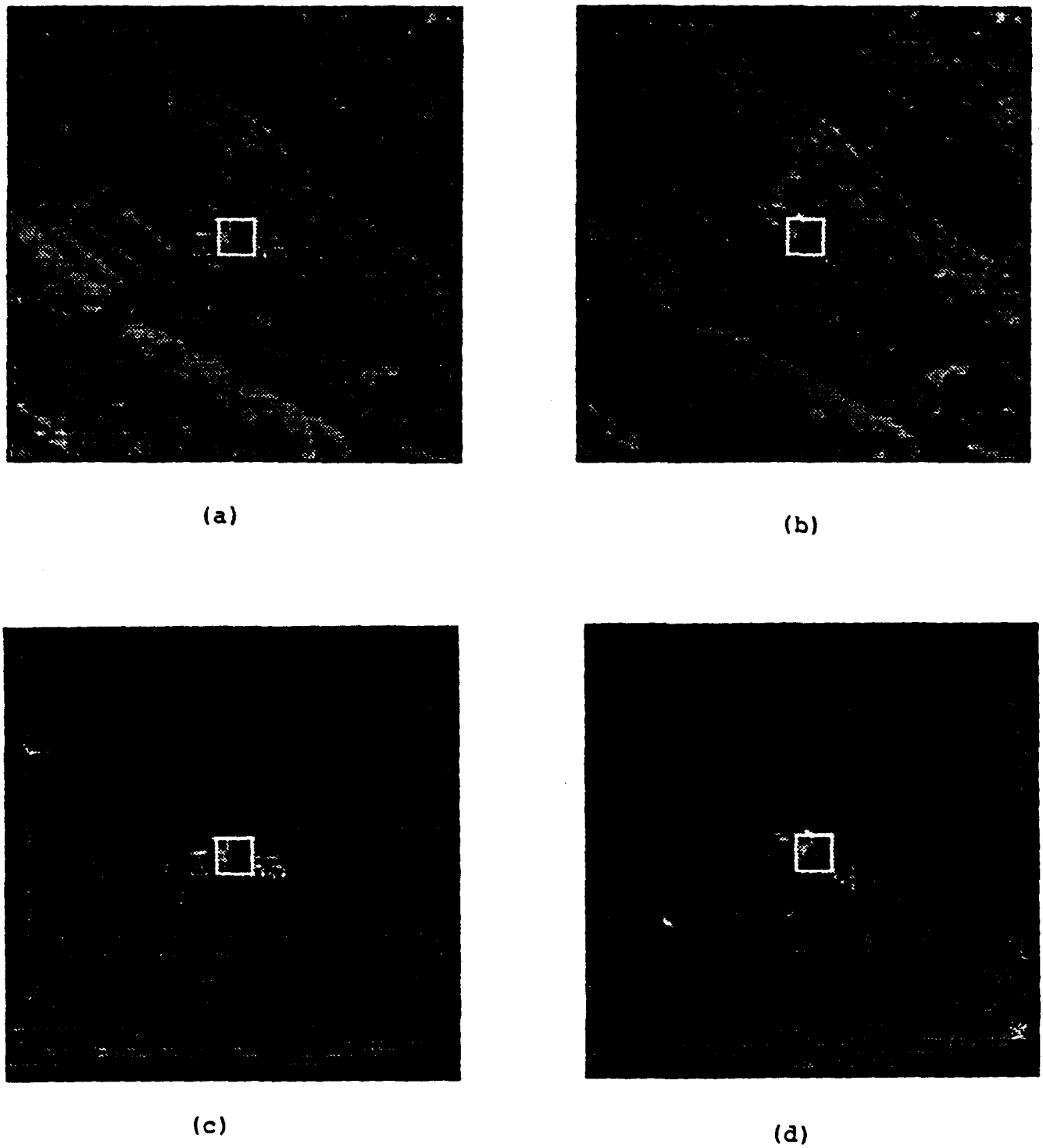
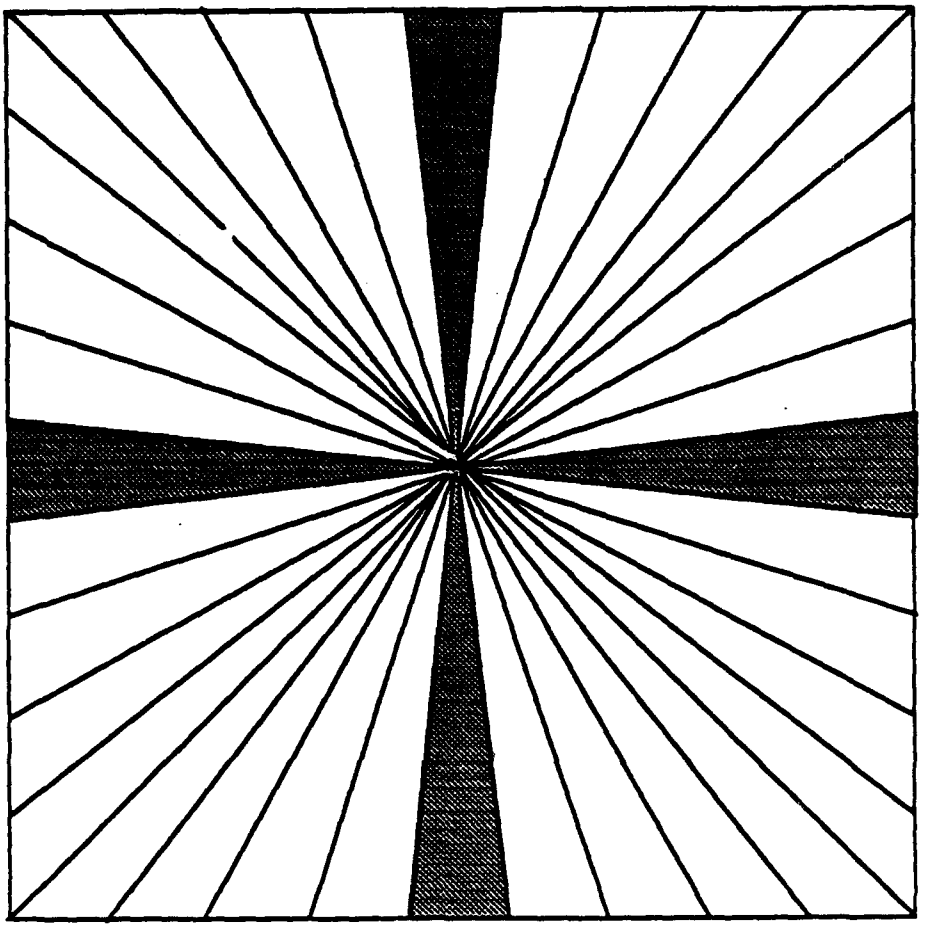
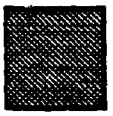


Figure 5.3 A 9 x 9 pixel grid superpixel on (a) truck on city70 background at a rotation of 0 degrees, (b) truck on city70 background at a rotation of 20 degrees, (c) truck on trbk70 background at a rotation of 0 degrees, and (d) truck on trbk70 background at a rotation of 20 degrees.





— Wedge not used for neural network inputs

Figure 5.4 **Formation of 32 wedges for neural network inputs from a 9 x 9 pixel grid.**

6. NEURAL NETWORK DESIGNS

6.1 Neural network architecture and training

Neural computing attempts to use architectures and processing techniques similar to those found in biological neural systems. Biological brains store and learn information using cells called neurons. Each neuron has associated with it input dendrites and an output axon. The input dendrites receive chemical stimuli through a synapse connection from many other neurons by means of their respective axons. If enough total stimulus is present, the neuron "fires" by releasing a signal along its axon. The strength of this signal is determined by the incoming stimuli to the dendrites. The reactions in biological brains are chemical, but they have electrical side effects which can be measured. Learning is accomplished by adapting the strength of the signal carried along the axon to other neurons. Memory is achieved by storing the strengths or weights of the neuron interconnections. Modeling neurons requires multiple variable inputs (to simulate the dendrites), a transfer function (to simulate the neural firing threshold), and multiple variable outputs (to simulate axon strengths connected to other neurons). By interconnecting such model neurons it is possible to simulate processes similar to those accomplished by biological brains. These processes are extremely parallel and are unlike the typical Von Neuman processes which are the basis

for modern digital computing. Digital computing is often slow and inappropriate for solving problems such as hitting a moving ball with a bat or backing up a 18 wheel semi-trailer truck to a loading dock.

In general, neural networks are used to establish input/ output relationships that are not easily described by rules. If rules can be identified for mapping the input to the output, then digital computing should be applicable. Neural networks generate their own rules by learning from examples. The neural networks employed here use supervised learning and require adaptation of the neuron interconnection weights.

6.2. Back-propagation neural network

The back-propagation neural network illustrated in Figure 6.1 uses a learning algorithm based on reducing the error between the actual output of the network and the desired output. Error reduction is accomplished by modifying the neuron interconnection weights. A back-propagation neural network has at least three layers: an input layer, an output layer, and a "hidden" layer typically not connected to any inputs or outputs. It is feed-forward, which means that the outputs from any layer are never fed back to previous layers. The back-propagation neural network uses delta-rule learning, which is a gradient decent procedure that adjusts the interconnection weights by minimizing the sum of the squared differences between the actual neural network output and the desired output. For an output layer of k neurons this function is

$$E = \frac{1}{2}\sum_{k}(d_k - y_k)^2$$

where d_k is the desired output of the k th neuron

y_k is the actual output of the k th neuron.

The back-propagation neural network uses supervised learning, which means that d_k is known. The value of y_k is

$$y_k = f(z_k)$$

where f(v) is typically the sigmoidal transfer function

$$f(v) = \frac{1}{1 + e^{-v}}$$

The first derivative of this function is

$$f'(v) = f(v)\,[1 - f(v)].$$

A typical argument of this function is

$$z_k = \sum_{j} w_{jk}\, y_j$$

where w_{jk} is the interconnection weight of the j th hidden neuron to the k th output neuron

y_j is the value of the j th hidden layer neuron.

y_j is given by

$$y_j = f(z_j)$$

where z_j is the sum of the weighted outputs of the input layer or

$$z_j = \sum_i w_{ij} y_j$$

where w_{ij} is the interconnection weight of the i th input neuron to the j th hidden neuron.

y_i is the i th input.

Initially the interconnection weights are set to small random values. To minimize the sum of squared differences between the actual neural network outputs and the desired outputs, a delta weight is determined. For hidden-to-output layer weights the change in weights as given in Figure 6.2 is

$$\Delta w_{jk} = \eta \, \delta_k \, y_j$$

where η is a gain constant which controls the strength of the weight change

δ_k is defined in Figure 6.2.

For input-to-hidden layer weights the change in weights as given in Figure 6.2 is

$$\Delta w_{ij} = \eta \left(\sum_k \delta_k \, w_{jk} \right) f'(z_j) \, y_j$$

Figure 6.2 illustrates the gradient descent technique used to adjust the interconnection weights. It is advantageous to increase the learning rate by

adding a momentum term to the delta weights [Rumenhart 1986]. For hidden-to-output layer weights the new delta weight with momentum is

$$\Delta w_{jk}\,(n+1) = \eta\ \delta_k\ y_j + \alpha\ \Delta w_{jk}\,(n)$$

where n is the iteration number

α is the momentum constant, which controls the strength of the weight change in terms of past weight changes.

For input-to-hidden layer weights the new delta weight with momentum is

$$\Delta w_{ij}\,(n+1) = \eta \left(\sum_k \delta_k\ w_{jk} \right) f'(z_j)\ y_j + \alpha\ \Delta w_{ij}\,(n)$$

Presenting enough representative examples of the known input/ output behavior establishes the patterns used to interpolate or approximate the outputs for inputs not used in training [Lipmann 1987]. These patterns are stored in the interconnection weights.

6.3. Stretch and hammer neural network

The stretch and hammer neural network illustrated in Figure 6.3 is designed to interpolate the training data by fitting a continuous hyper-function of dimensionality equal to the number of inputs. This network is used for filter synthesis as follows. First, the input space is "stretched" into the principal component space. For the 138 input vectors of 32 values each used to train the network, a 138 X 32 training data matrix is formed and pre-multiplied by its transpose:

$$\begin{pmatrix} x_1^1 & x_1^2 & \dots & x_1^{138} \\ x_2^1 & x_2^2 & \dots & x_2^{138} \\ \dots & & & \\ x_{32}^1 & x_{32}^2 & \dots & x_{32}^{138} \end{pmatrix} \begin{pmatrix} x_1^1 & x_2^1 & \dots & x_{32}^1 \\ x_1^2 & x_2^2 & \dots & x_{32}^2 \\ \dots & & & \\ x_1^{138} & x_2^{138} & \dots & x_{32}^{138} \end{pmatrix} = \begin{pmatrix} m_1^1 & m_1^2 & \dots & m_1^{32} \\ m_2^1 & m_2^2 & \dots & m_2^{32} \\ \dots & & & \\ m_{32}^1 & m_{32}^2 & \dots & m_{32}^{32} \end{pmatrix},$$

where x_i^j is the i th component of the j th training example with each component scaled so that its mean (for all training examples) is zero. The eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_{32}$ of this matrix are obtained as the solutions of

$$\det \begin{pmatrix} m_1^1-\lambda & m_1^2 & \dots & m_1^{32} \\ m_2^1 & m_2^2-\lambda & \dots & m_2^{32} \\ \dots & & & \\ m_{32}^1 & m_{32}^2 & \dots & m_{32}^{32}-\lambda \end{pmatrix} = 0.$$

Solving the simultaneous equations

$$\begin{pmatrix} m_1^1 & m_1^2 & \dots & m_1^{32} \\ m_2^1 & m_2^2 & \dots & m_2^{32} \\ \dots & & & \\ m_{32}^1 & m_{32}^2 & \dots & m_{32}^{32} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \dots \\ u_{32} \end{pmatrix} = \lambda_i \begin{pmatrix} u_1 \\ u_2 \\ \dots \\ u_{32} \end{pmatrix} \quad , \quad i = 1, 2, \dots, 32,$$

for each eigenvalue yields each orthogonal 32-dimensional principal components axis (with components u_i), where each axis is scaled in units of its eigenvalue. Next, a least squares hyper-plane is fit to these points using

$$\begin{pmatrix} 1 & 1 & \dots & 1 \\ u_1^1 & u_2^1 & \dots & u_{138}^1 \\ u_1^2 & u_2^2 & \dots & u_{138}^2 \\ \dots & & & \\ u_1^{32} & u_2^{32} & \dots & u_{138}^{32} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ \dots \\ z_{138} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ u_1^1 & u_2^1 & \dots & u_{138}^1 \\ u_1^2 & u_2^2 & \dots & u_{138}^2 \\ \dots & & & \\ u_1^{32} & u_2^{32} & \dots & u_{138}^{32} \end{pmatrix} \begin{pmatrix} 1 & u_1^1 & u_1^2 & \dots & u_1^{32} \\ 1 & u_2^1 & u_2^2 & \dots & u_2^{32} \\ \dots & & & & \\ 1 & u_{138}^1 & u_{138}^2 & \dots & u_{138}^{32} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \dots \\ b_{32} \end{pmatrix}$$

where u_i^j represent the inputs in the zero-mean principal component space, z_j are the outputs for these inputs, b_0 is the intercept and the remaining b_i's are the slopes of the 32-dimensional hyper-plane. The b_i are determined by solving

the above simultaneous equations. The least squares hyper-plane will not in general match any of the outputs. To produce a perfect match at the outputs, a radial Gaussian "hammer" is used to deform the hyper-plane until it intersects each output value. This deformation is accomplished using the following matrix algebra:

$$\begin{pmatrix} (z_1 - z_1') \\ (z_2 - z_2') \\ \cdots \\ (z_{138} - z_{138}') \end{pmatrix} = \begin{pmatrix} f_1^1 & f_1^2 & \cdots & f_1^{138} \\ f_2^1 & f_2^2 & \cdots & f_2^{138} \\ \cdots & & & \\ f_{138}^1 & f_{138}^2 & \cdots & f_{138}^{138} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \cdots \\ c_{138} \end{pmatrix},$$

where $f_i^j = \exp\{[-1/(2s_j)][(u_j^1 - u_i^1)^2 + (u_j^2 - u_i^2)^2 + \ldots + (u_j^{32} - u_i^{32})^2]\}$ for i, j = 1, 2, ..., 138, z_i' are the outputs given by the least squares hyper-plane, z_i are the known training outputs, s_i are the standard deviations of the radial Gaussians, and c_i are the radial Gaussian hammer weights. The diagonal elements of this matrix are unity, and the standard deviations of the Gaussian hammers are selected such that the off-diagonal elements of each column add to a value slightly less than unity. The c_i may be obtained using Gaussian elimination, singular value decomposition, or any of several other techniques for solving simultaneous linear equations. The stretch and hammer neural network is designed to exactly match every training output and to smoothly generalize for all other outputs.

6.4 Comparison of stretch and hammer and back-propagation

There are important differences in the architectures of the stretch and hammer and the back-propagation neural networks. The stretch and hammer neural network has a number of adjustable parameters that depends on the number of training examples. The adjustable parameters for a back-propagation neural network depend on the number of neurons employed. A back-propagation neural network with k input neurons, m hidden layer neurons, n output neurons, and no interlayer interconnections has km + mn + m + n adjustable parameters. A stretch and hammer neural network has n(k + 1 + p) adjustable parameters, where p is the number of examples used for training. A 31 input, 200 neuron hidden layer, 600 output back-propagation neural network with 50% of the interconnections randomly removed, as was used for this research, has

$$0.5 * (31 * 200 + 200 * 600 + 200 + 600) = 63{,}500$$

adjustable parameters. This value is independent of the number of training examples. Using 138 training examples, as was the case for this research, a stretch and hammer neural network has

$$600 * (31 + 1 + 138) = 102{,}000$$

adjustable parameters.

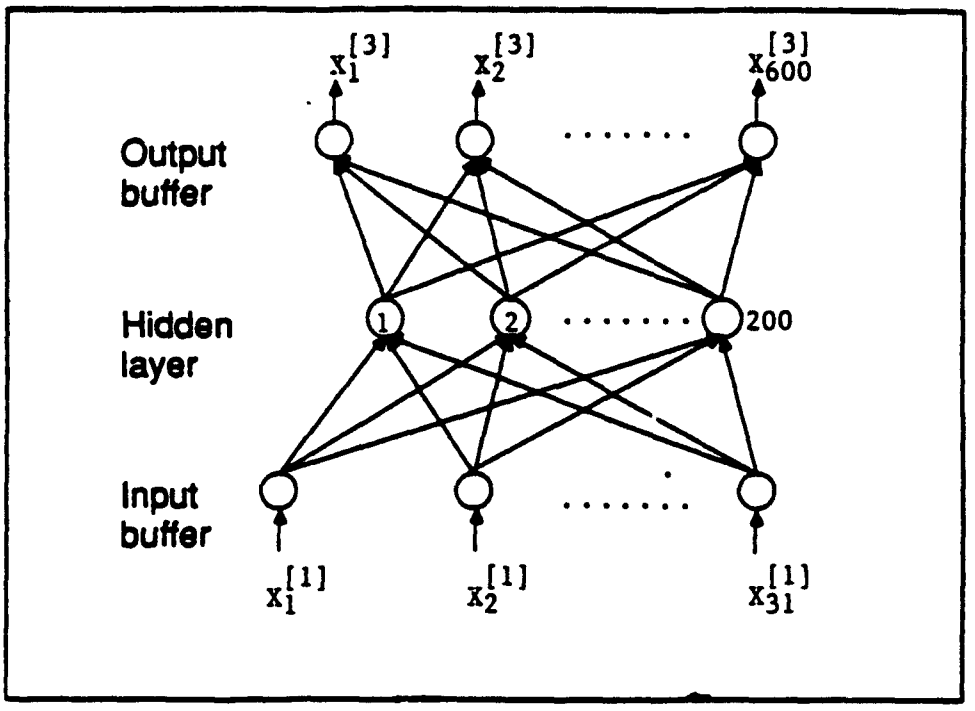
To compare performance of the two networks, a simple exercise involving rotation estimation rather than filter synthesis was carried out for which both networks had the same number of adjustable parameters. A training set of 5 examples with one input and one output was used, and the back-propagation

neural network had two hidden neurons. The number of adjustable parameters for both neural networks was therefore

$$1*(1 + 1 + 5) = (1*2 + 2*1 + 2 + 1) = 7.$$

The input was the average gray level value for one quadrant of a 9 x 9 pixel grid, and the output was the in-plane angular rotation of the target. The examples used for training were at rotation angles of 0, 8, 16, 24, and 30 degrees, and testing results were obtained at angles of 0, 1, ..., 30 degrees.

The back-propagation neural network results varied as the learning process progressed. This variation is illustrated in Figure 6.4, which shows the best performance for an RMS training error of 3.5%. After training the stretch and hammer neural network yielded approximately .5 degree average error for testing angles. At 3.5% RMS training error, the back-propagation neural network testing results approached those of the stretch and hammer neural network as illustrated in Figure 6.5. When the back-propagation neural network was allowed to continue training to an RMS training error of approximately 0% , the results show a larger error for testing, although at the training examples the error is near zero as illustrated in Figure 6.6. Thus the back-propagation neural network does well at retrieving the training examples but poorly at interpolating the testing examples. The stretch and hammer neural network retrieves the training examples exactly by definition, and in the above exercise interpolates better than the back-propagation neural network for the testing examples.



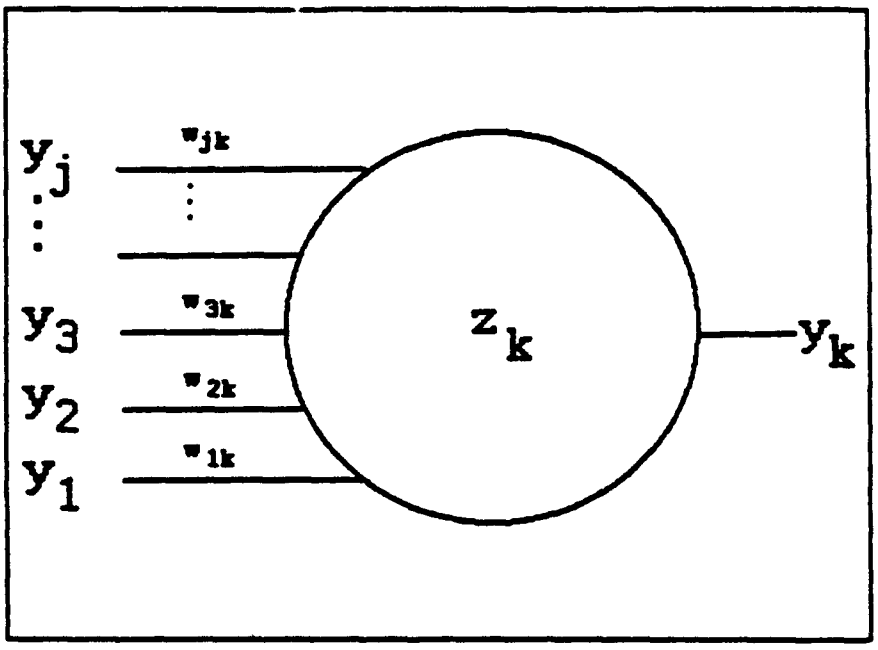


Figure 6.1 (a) A back-propagation neural network with 31 inputs, 200 hidden-layer neurons, and 600 output neurons. (b) simple processing element where w_{ij} - weighted connections, k - neuron number in current layer, j - neuron number of previous layer.

change of hidden-to output layer weights

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}}$$

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial z_k} \frac{\partial z_k}{\partial w_{jk}}$$

$$\frac{\partial z_k}{\partial w_{jk}} = y_j$$

$$\frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial z_k}$$

$$\frac{\partial E}{\partial y_k} = d_k - y_k$$

$$\frac{\partial y_k}{\partial z_k} = f'(z_k)$$

$$\delta_k = (y_k - d_k)\ f'(z_k)$$

$$\Delta w_{jk} = \eta\, \delta_k\, y_k$$

change of input-to-hidden layer weights

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial z_j} \frac{\partial z_j}{\partial w_{ij}}$$

$$\frac{\partial z_j}{\partial w_{ij}} = y_i$$

$$\frac{\partial E}{\partial z_j} = \sum_k \frac{\partial E}{\partial z_k} \frac{\partial z_k}{\partial z_j}$$

$$\frac{\partial E}{\partial z_k} = -\delta_k$$

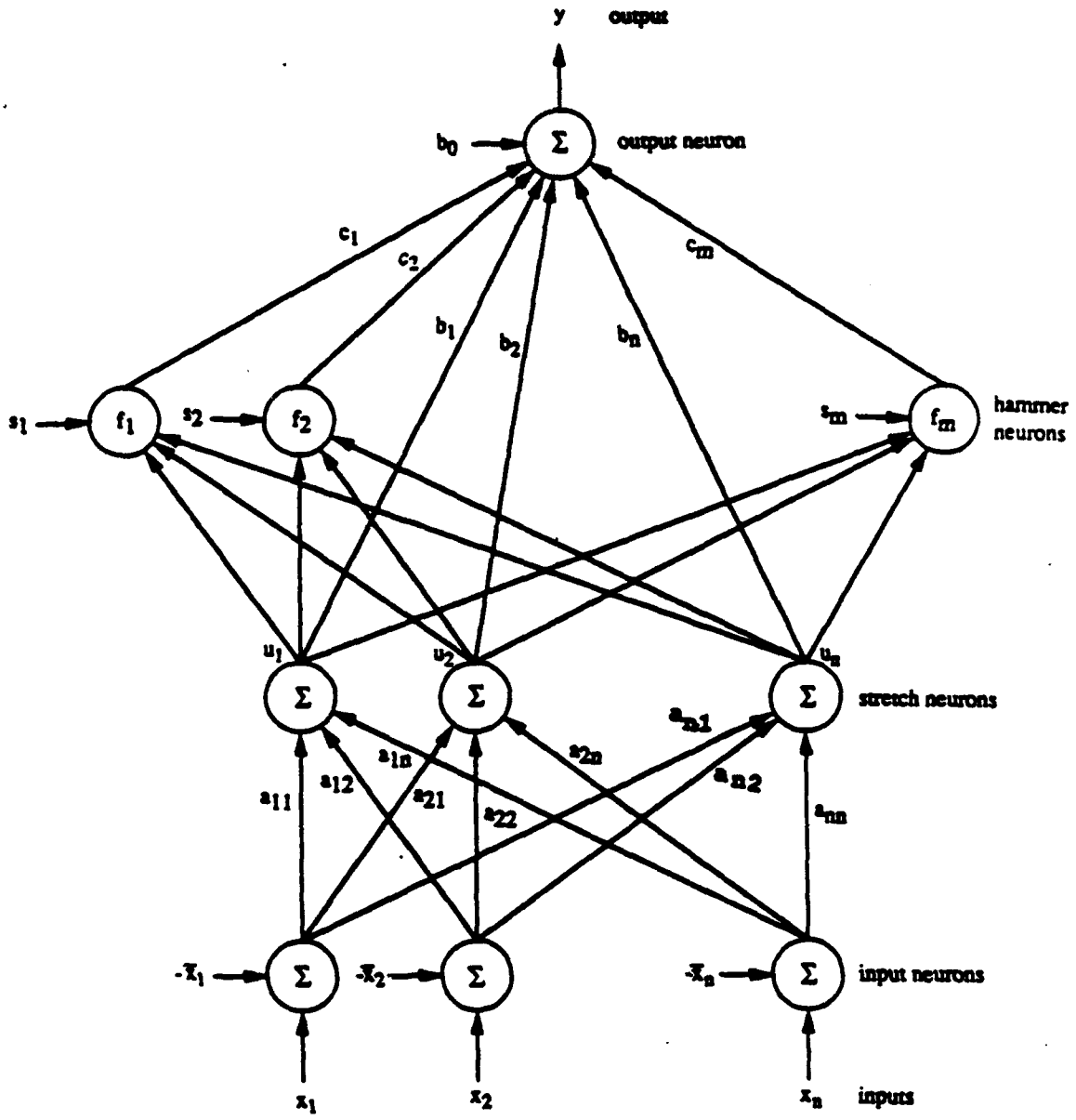
$$\frac{\partial z_k}{\partial z_j} = \frac{\partial z_k}{\partial y_j} \frac{\partial y_j}{\partial z_j}$$

$$\frac{\partial z_k}{\partial y_j} = w_{jk}$$

$$\frac{\partial y_j}{\partial z_j} = f'(z_j)$$

$$\Delta w_{ij} = \eta \left(\sum_k \delta_k\, w_{jk} \right) f'(z_j)\ y_i$$

Figure 6.2 Gradient descent technique used to determine changes in the interconnection weights.



$$y = b_0 + b_1u_1 + \ldots + b_nu_n + c_1f_1 + c_2f_2 + \ldots + c_mf_m$$

$$f_i = f_i(u_1, u_2, \ldots, u_n, s_i)\ ,\ i = 1, 2, \ldots, m$$

$$u_j = a_{j1}(x_1 - \bar{x}_1) + a_{j2}(x_2 - \bar{x}_2) + \ldots + a_{jn}(x_n - \bar{x}_n),\ j = 1, 2, \ldots, n$$

$$f_i = \exp[-(u_1^2 + u_2^2 + \ldots + u_n^2)/2s_i^2] \quad \text{radial Gaussian}$$

Figure 6.3 Stretch and hammer neural network with m examples and n inputs.

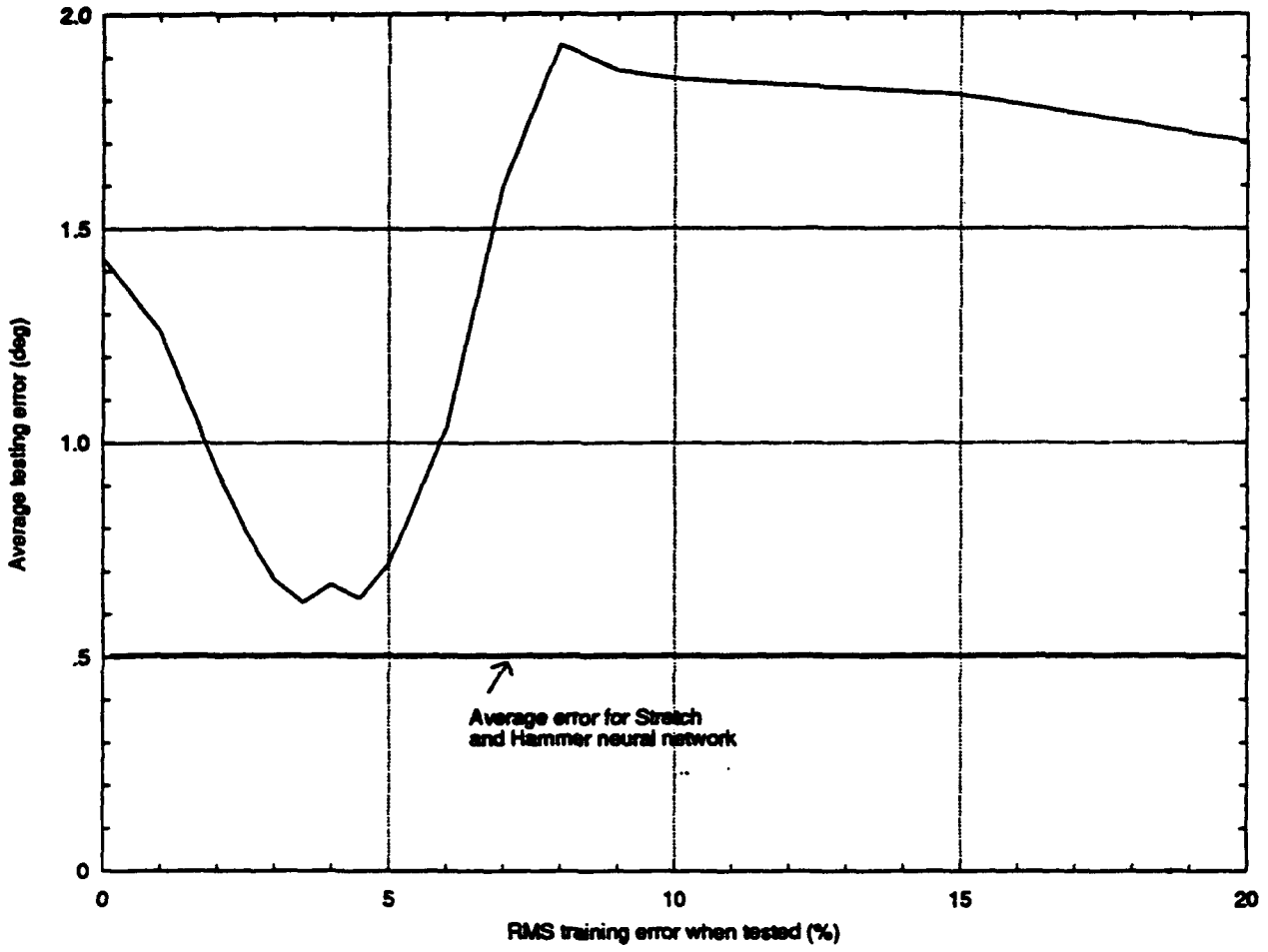


Figure 6.4 Comparison of average error (deg) for target rotation estimation. The back-propagation neural network was trained to various RMS training error values. The stretch and hammer neural network average error for testing was approximately 0.5 degrees.

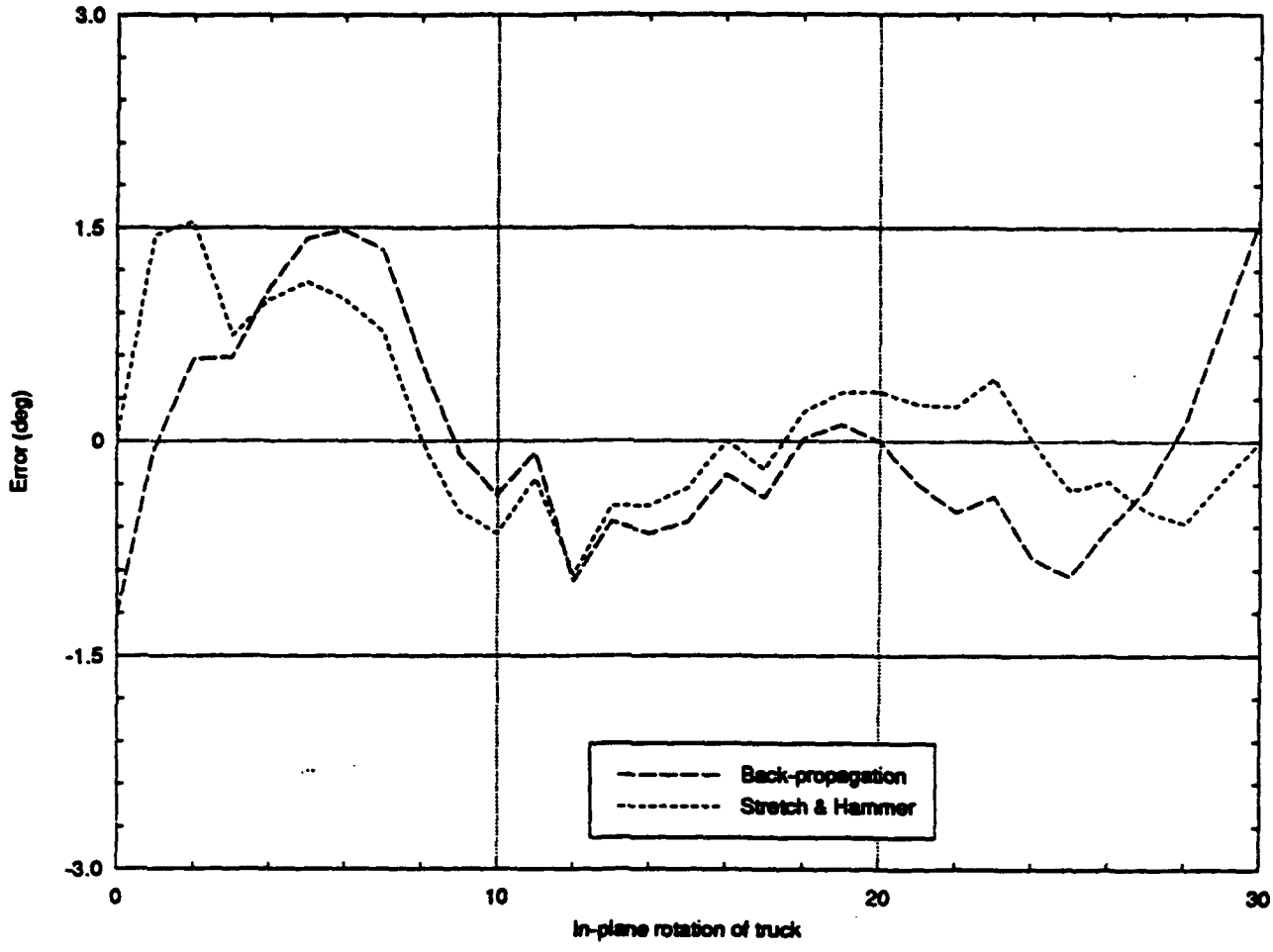


Figure 6.5 Comparison of error (deg) for target rotation estimation for a back-propagation RMS training error of 3.5% with stretch and hammer results.

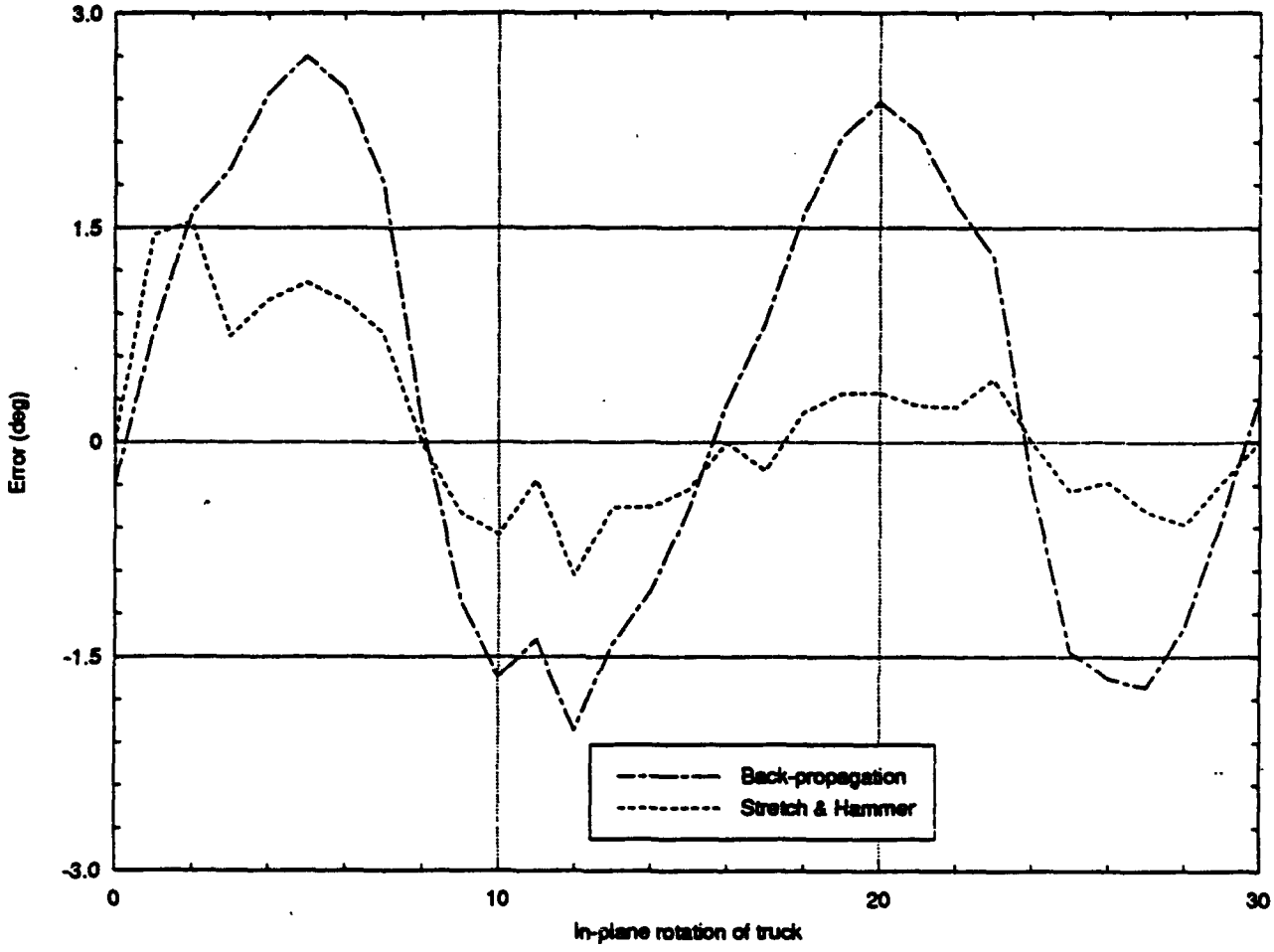


Figure 6.6 Comparison of error (deg) for target rotation estimation for a back-propagation RMS error of 0% with stretch and hammer results.

7. FILTER SYNTHESIS RESULTS

The results of four neural network filter synthesis cases are discussed in this section. The neural network outputs for each case are the 600 complex filter values described in Section 4. The first case uses a back-propagation neural network and 25 intensity values from a 5 x 5 target-centered pixel grid as inputs. The second case uses a back-propagation neural network and 25 computed values from a 9 x 9 target-centered pixel grid as inputs as described in Section 5. The third case uses a back-propagation neural network and 32 computed values from a 9 x 9 target-centered pixel grid as inputs as described in Section 5. The fourth case uses a stretch and hammer neural network and the same inputs as the third case. The networks were tested by varying one or both of two input scene parameters: the angle of target rotation and the type of cluttered background. For the gray-level correlator input scenes the angles were offset by 2 degrees, and for the binarized correlator input scenes the angles were offset by 1 degree. Varying the angle of rotation permitted investigation of the performance of the network in interpolating between training angles. Varying the type of background permitted investigation of the robustness of the network. The goal was to train the network to ignore background effects.

7.1. Correlation peak metrics

Two correlation peak metrics were used to evaluate neural network performance: a peak-to-sidelobe ratio and a peak-to-clutter ratio. A ratio of less than 3 dB is inadequate for use in the HAC system. The correlation plane is expected to exhibit a peak near the target center. For all cases considered here the target was superimposed on the center of a background scene. For gray level correlator input scenes a peak finding algorithm was used to locate peaks in three regions of the correlation plane: region 1 was a target-centered 5 x 5 pixel grid, region 2 was a target-centered 15 x 15 pixel grid excluding region 1, and region 3 was the remaining area of the 128 by 128 pixel correlation plane. For binary correlator input scenes a peak finding algorithm was used to locate peaks in two regions of the correlation plane: region 1 was a target-centered 5 x 5 pixel grid, and region 2′ was the remaining area of the correlation plane. A peak-to-sidelobe ratio measurement was made for gray-level correlator input scenes by comparing the highest peak in region 1 with the highest peak in region 2. A peak-to-clutter ratio measurement was made by comparing the highest peak in region 1 with the highest peak in region 3 for gray-level correlator input scenes and with the highest peak in region 2′ for binary correlator input scenes. Peak-to-sidelobe ratios were smaller than peak-to-clutter ratios for the gray level input scenes.

7.2. Back-propagation neural network results (5 x 5 grid, peak-to-sidelobe)

Using gray level correlator inputs and a TPAF in the filter plane, a back-propagation neural network with 25 inputs, 200 hidden neurons, and 600 outputs successfully synthesized filters. The hidden neurons had sigmoidal transfer functions, and the output neurons had summation transfer functions. The network was trained on a set of 180 input scenes of 128 by 128 pixels, 90 of which corresponded to the truck target rotated at angles of 0, 4, ..., 356 degrees on a uniform gray 127 intensity-level background. The remaining 90 scenes corresponded to the truck at angles of 0, 4, ..., 356 degrees on a city background. A single filter was produced for each rotation angle as described in Section 4. The filters were made from the binarized Fourier transform of the truck on a gray-level background only. Thus, there were only 90 output filter examples, one for each training rotation angle. Two backgrounds were used so that the neural network could be trained to ignore pixel values that were not on the target (i.e., that were background pixels).

Figures 7.1 through 7.5 show graphs of the peak-to sidelobe ratio (in dB) versus target rotation (in degrees) for a variety of cases. These results are summarized in Table 7.1. The inputs were the 25 intensity values from a target-centered 5 x 5 pixel grid. The outputs were the 600 coded values used to generate a 10-60 bandpass 3 x 3 superpixel TPAF. Since the peak-to-clutter ratio was always higher than the peak-to-sidelobe ratio, network performance assessment was based on the peak-to-sidelobe ratio.

The "zero degree filter" curve shows that more than one fixed filter is needed to accommodate target rotation. The filter for this curve was symmetric and was constructed from the binarized Fourier transform of the truck on a 127 gray-level background at a rotation angle of zero degrees. Performance for this filter depends on the background. It is apparent from Figures 7.1 through 7.5 that the peak-to-sidelobe ratio decreases rapidly for only a few degrees of rotation: target rotation by more than approximately 5 degrees decreases the correlation peak to the noise level.

The "best expected filter from network" curve shows the upper limit for the peak-to-sidelobe ratio because the trained-for filter is used to correlate with the input scene. Figures 7.1 through 7.5 show that the peak-to-sidelobe ratio varies as the target is rotated, but a relatively high value is typically maintained.

The "filter synthesized by network" curve shows performance when the filter synthesized by the network is used to correlate with the input scene. Performance depends on variations in scene parameters used to test the network and is summarized in Table 7.1.

In general, when the neural network was tested for backgrounds used in training, the network performed very well and almost matched the best expected performance. When the angle of target rotation was offset by 2 degrees the network performance degraded slightly. However, for the city background (which was used in training) the network performance was still close to the best expected performance. For this case only one input parameter

was varied, namely the rotation angle. As expected, when the background was also varied the network performance degradation increased, but performance was adequate for pattern recognition by optical correlation in all but 12 out of 270 testing cases. In these 12 cases the peak-to-sidelobe ratio was below the 3 dB line, which is not acceptable for use in the correlator.

Background	Angles of Rotation	Comments
Training-		
Gray	0, 4, ..., 360	The performance of the network matches the best expected performance almost exactly.
City	0, 4, ..., 360	The performance of the network matches the best expected performance almost exactly.
Testing-		
City	2, 6, ..., 358	The general performance of the network follows the best expected performance. Performance drops below the 3 dB line for rotation angles of 66, 242, 258, and 322 degrees.
Bushy	2, 6, ..., 358	The performance of the network is slightly degraded. Performance drops below the 3 dB line for rotation angles of 130 and 310 degrees.
Newbk80	2, 6, ..., 358	The newbk80 background camouflages the truck and a reduction in the peak-to-sidelobe ratio is expected. The overall performance, however, indicates acceptable peak-to-sidelobe ratios for target recognition. Performance drops below the 3 dB line for rotation angles of 78, 96, 118, 122, 130, and 210 degrees.

Table 7.1 Shows the performance of a filter synthesized by a back-propagation neural network using a 5 x 5 sampling grid and the peak-to-sidelobe metric for a truck rotated on a variety of backgrounds.

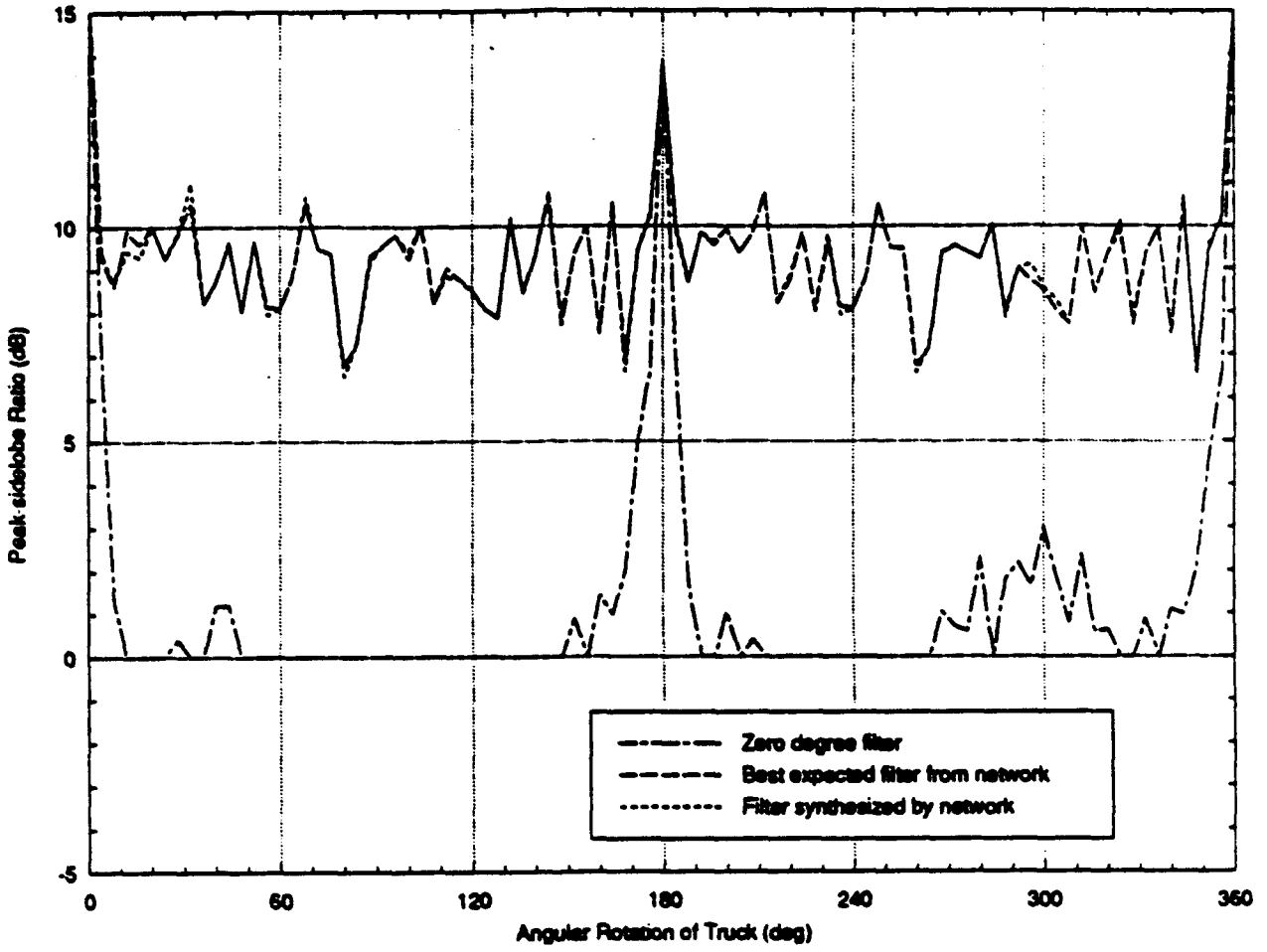


Figure 7.1 **Filter synthesis results using a back-propagation neural network for the truck on the gray background at truck rotation angles of 0, 4, ..., 360 degrees. The network was trained on both gray and city backgrounds at truck rotation angles of 0, 4, ..., 360 degrees using as inputs 25 intensity values from a target-centered 5 x 5 pixel grid.**

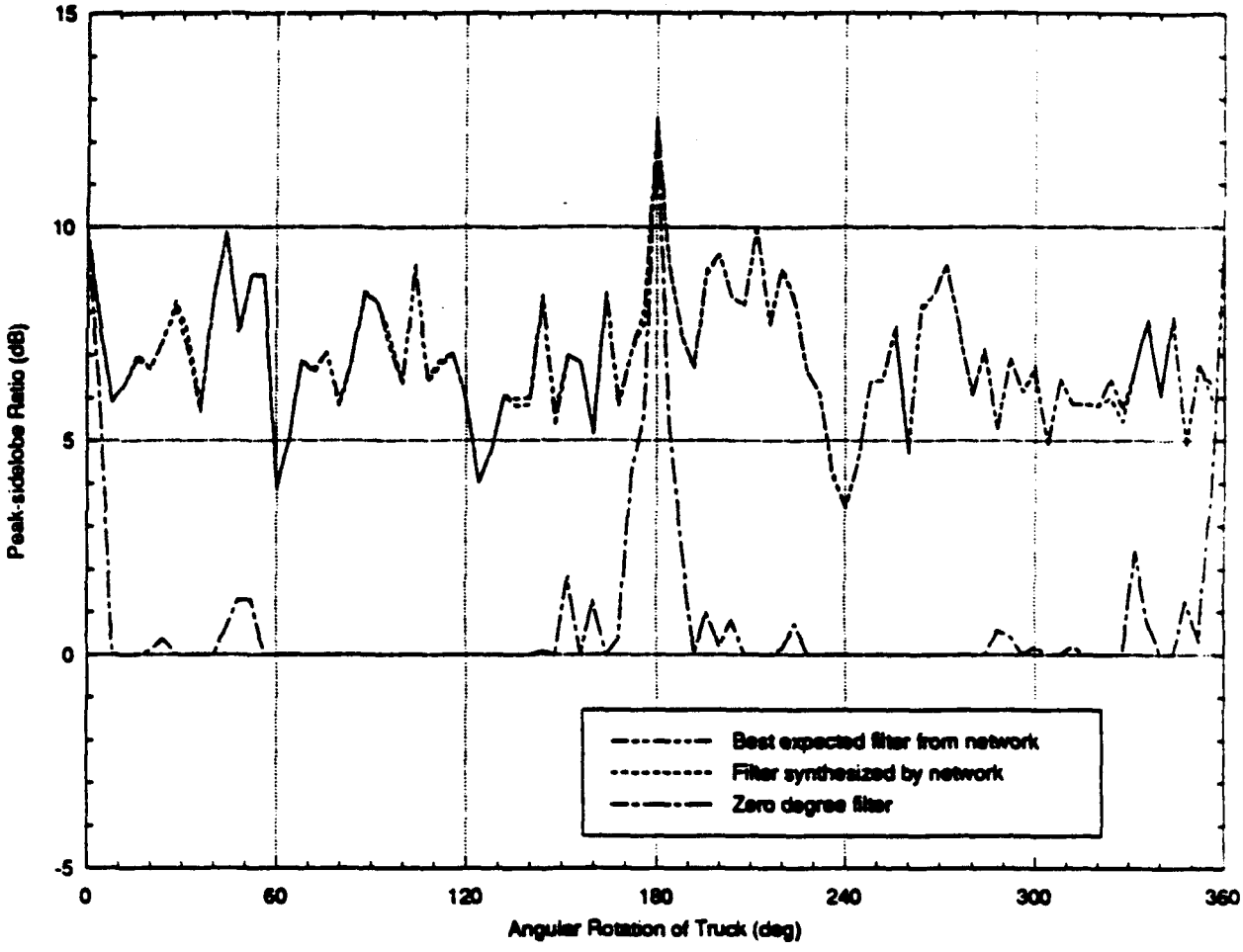


Figure 7.2 Filter synthesis results using a back-propagation neural network for the truck on the city background at truck rotation angles of 0, 4, ..., 360 degrees. The network was trained on both gray and city backgrounds at truck rotation angles of 0, 4, ..., 360 degrees using as inputs 25 intensity values from a target-centered 5 x 5 pixel grid.

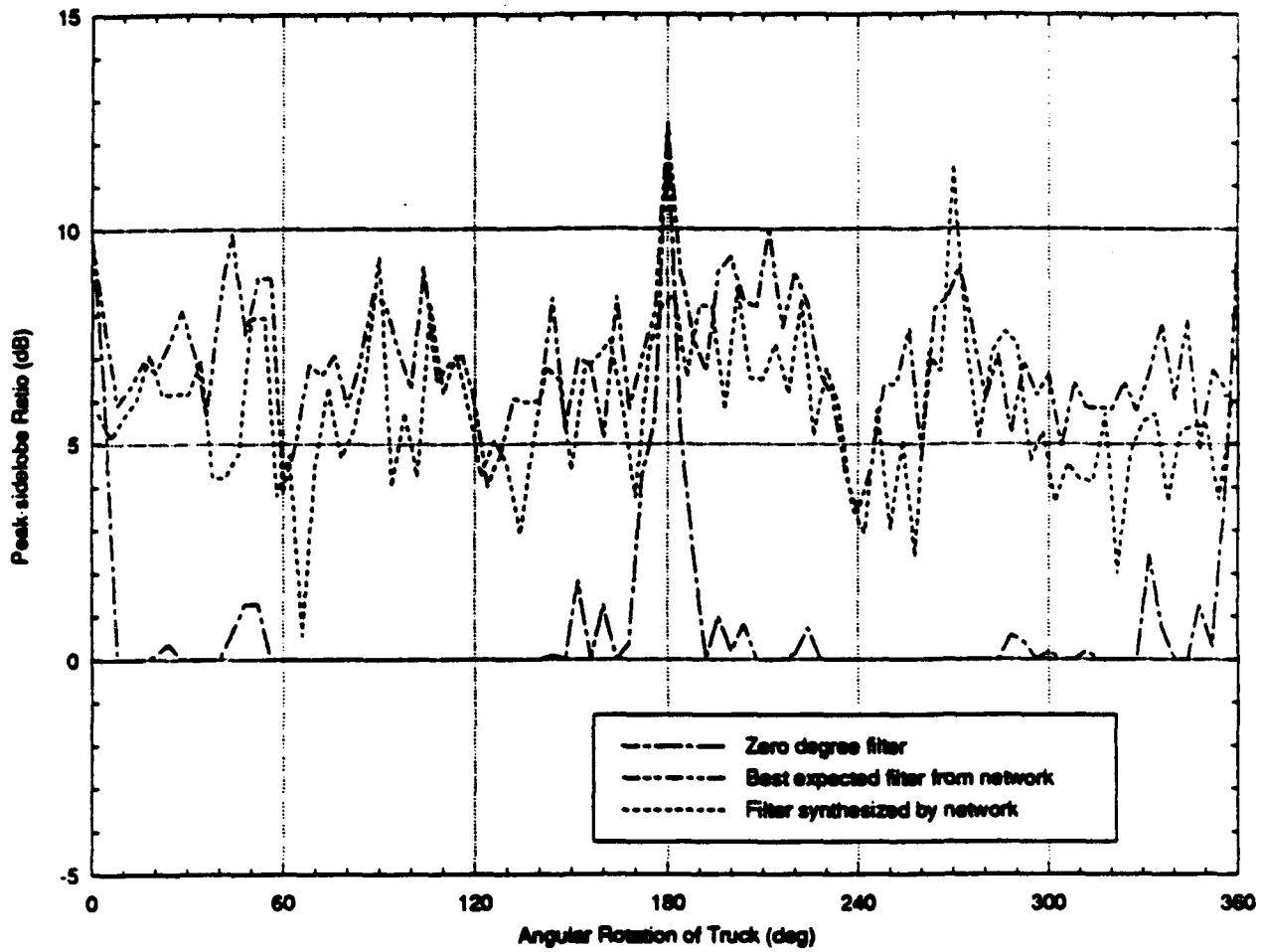


Figure 7.3 Filter synthesis results using a back-propagation neural network for the truck on the city background at truck rotation angles of 2, 6, ..., 358 degrees. The network was trained on both gray and city backgrounds at truck rotation angles of 0, 4, ..., 360 degrees using as inputs 25 intensity values from a target-centered 5 X 5 pixel grid.

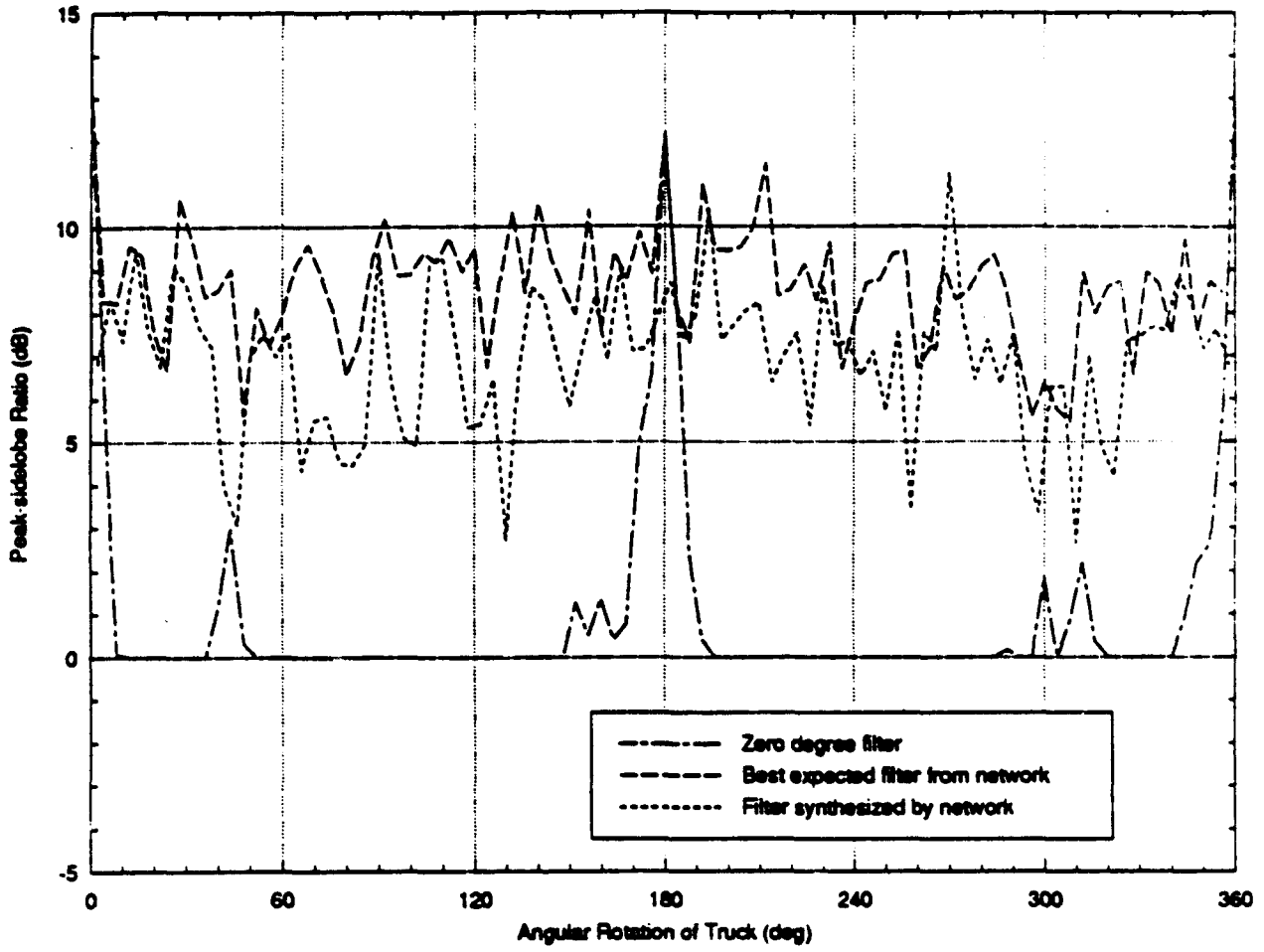


Figure 7.4 Filter synthesis results using a back-propagation neural network for the truck on the bushy background at truck rotation angles of 2, 6, ..., 358 degrees. The network was trained on both gray and city backgrounds at truck rotation angles of 0, 4, ..., 360 degrees using as inputs 25 intensity values from a target-centered 5 x 5 pixel grid.

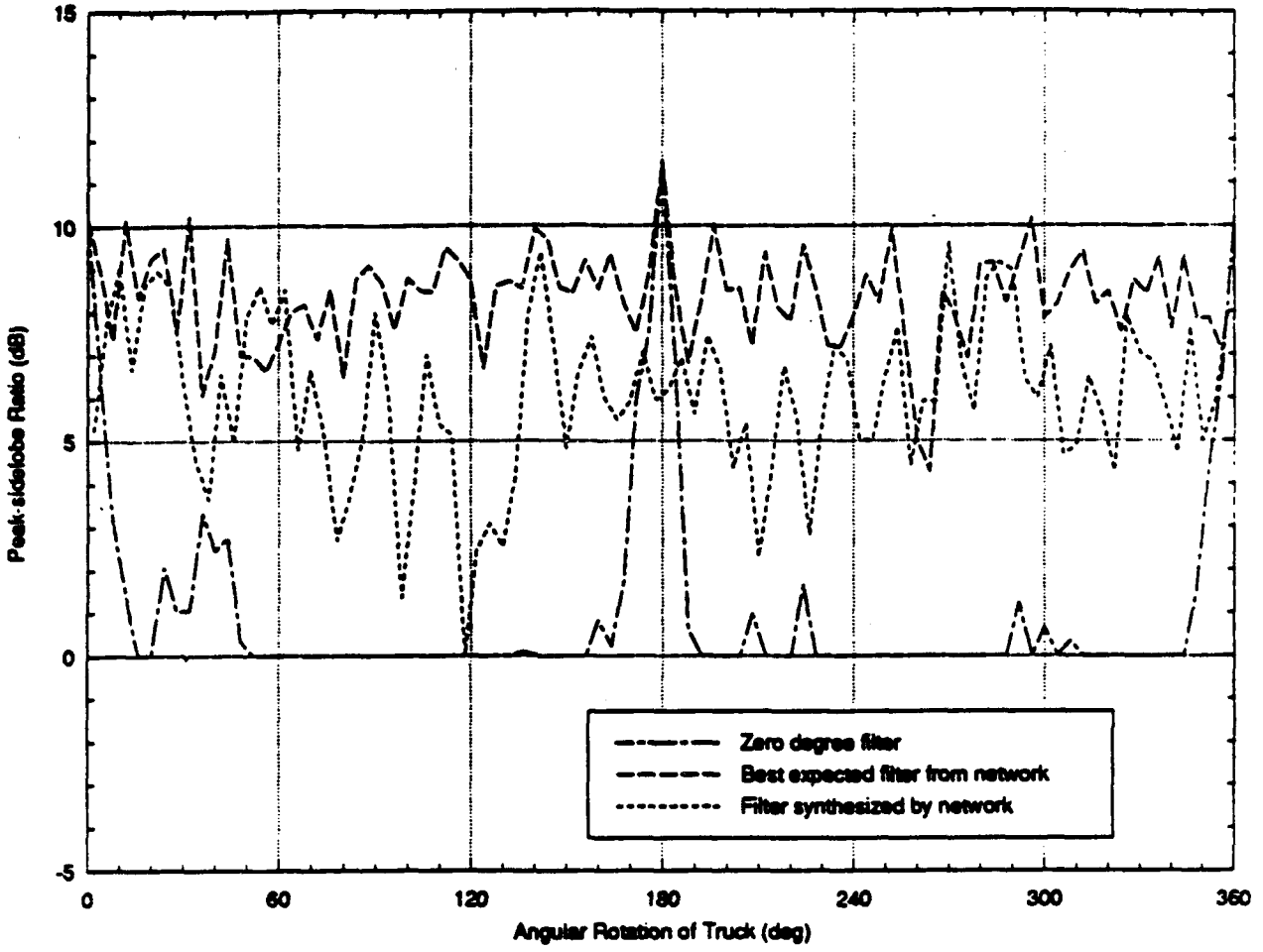


Figure 7.5 **Filter synthesis results using a back-propagation neural network for the truck on the newbk80 background at truck rotation angles of 2, 6, ..., 358 degrees. The network was trained on both gray and city backgrounds at truck rotation angles of 0, 4, ..., 360 degrees using as inputs 25 intensity values from a target-centered 5 x 5 pixel grid.**

7.3. Back-propagation neural network results (9 x 9 grid, peak-to-clutter)

Using binarized correlator inputs and a TPAF in the filter plane, a back-propagation neural network with 25 inputs (computed from the gray-level input image), 200 hidden neurons, and 600 outputs successfully synthesized filters. The neural network was trained on a set of 138 input scenes of 128 by 128 pixels, 46 of which corresponded to the truck rotated at angles of 0, 2, ..., 90 degrees on a uniform gray 66 intensity-level background, 46 of which corresponded to the truck rotated at angles of 0, 2, ..., 90 degrees on a uniform gray 142 intensity-level background, and the remaining 46 of which corresponded to the truck rotated at angles of 0, 2, ..., 90 degrees on a city background. A single filter was produced for each rotation angle as described in Section 4. The filters were made from the binarized Fourier transform of the truck on a blank background only. Thus, there were only 46 output filter examples, one for each training angle. Three backgrounds were used so that the neural network could be trained to ignore pixel values that were not on the target.

Figures 7.6 through 7.13 show graphs of the peak-to-clutter ratio (in dB) versus target rotation (in degrees) for a variety of cases. These results are summarized in Table 7.2. The inputs were 25 values from a target-centered 9 x 9 pixel grid as described in Section 5. The outputs were the 600 coded values used to generate a 10-60 bandpass 3 x 3 superpixel TPAF. The "zero degree filter", "best expected filter", and "filter synthesized by network" curves have

the same meaning as in Section 7.2. Performance depends on variations in scene parameters used to test the network and is summarized in Table 7.2.

When tested on the city70 background at angles of 0, 2, ..., 90 degrees (which were used in training) the neural network performance almost matched the best expected performance. When different backgrounds were used at the training rotation angles, the network performance degraded slightly. As expected, at testing rotation angles (1, 3, ..., 89 degrees) the network performance degradation increased. However, network performance was adequate for pattern recognition (above the 3 dB line) by optical correlation in all but 7 out of 318 testing cases.

Background	Angles of Rotation	Comments
Training-		
City70	0, 2, ..., 90	The performance of the network matches the best expected performance almost exactly.
Testing		
City70	1, 3, ..., 89	The general performance of the network follows the best expected performance. Performance drops below the 3 dB line. For target rotation angles of 25, 37, and 83 degrees.
Trbk70	0, 2, ..., 90	The general performance of the network closely follows the best expected performance.
Trbk70	1, 3, ..., 89	The performance of the network is adequate for pattern recognition.
Newbk80	0, 2, ..., 90	The general performance of the network closely follows the best expected performance.
Newbk80	1, 3, ..., 89	The general performance of the network follows the best expected performance. Performance drops below the 3 dB line for a rotation angle of 25 degrees.
Bushy	0, 2, ..., 90	The general performance of the network closely follows the best expected performance. Performance drops below the 3 dB line for a rotation angle of 66 degrees.
Bushy	1, 3, ..., 89	The general performance of the network closely follows the best expected performance. Performance drops below the 3 dB line for rotation angles of 65 and 81 degrees.

Table 7.2 Shows the performance of a filter synthesized by a back-propagation neural network using a 9 x 9 sampling grid and the peak-to-clutter metric for a truck rotated on a variety of backgrounds.

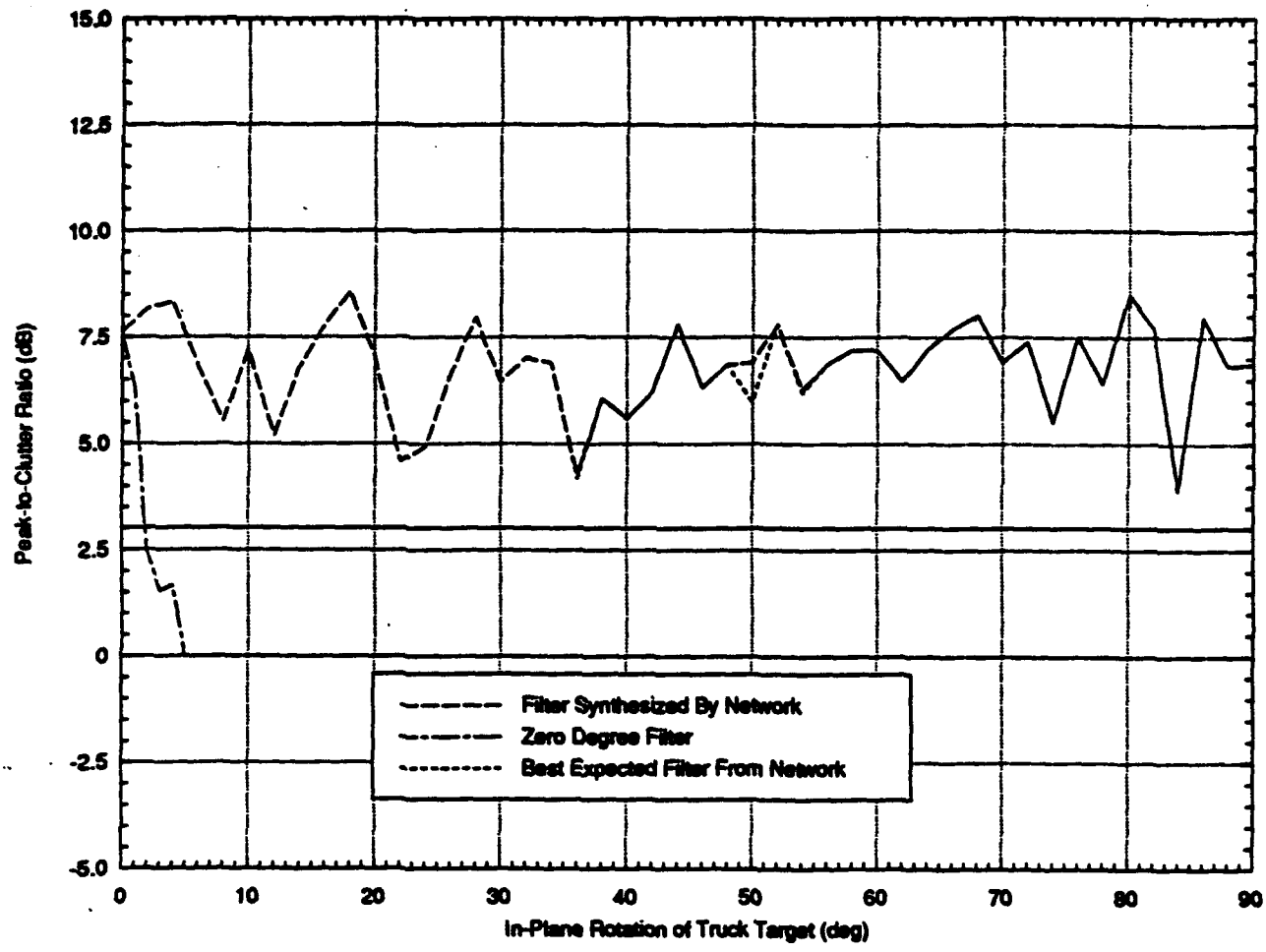


Figure 7.6 Filter synthesis results using a back-propagation neural network for the truck on the city70 background at truck rotation angels of 0, 2, ..., 90 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 25 computed values from a target-centered 9 x 9 pixel grid.

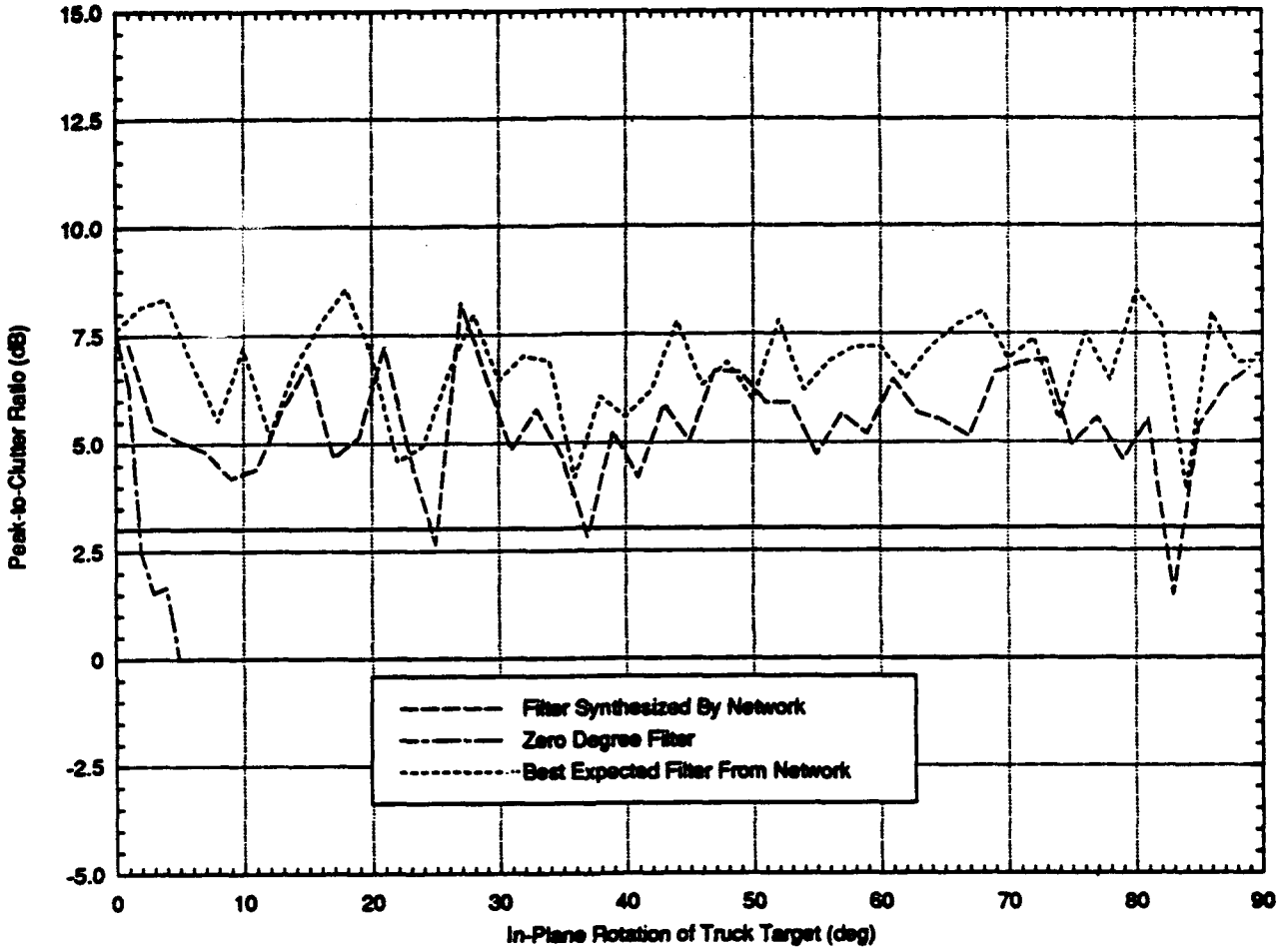


Figure 7.7 Filter synthesis results using a back-propagation neural network for the truck on the city70 background at truck rotation angels of 1, 3, ..., 89 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 25 computed values from a target-centered 9 x 9 pixel grid.

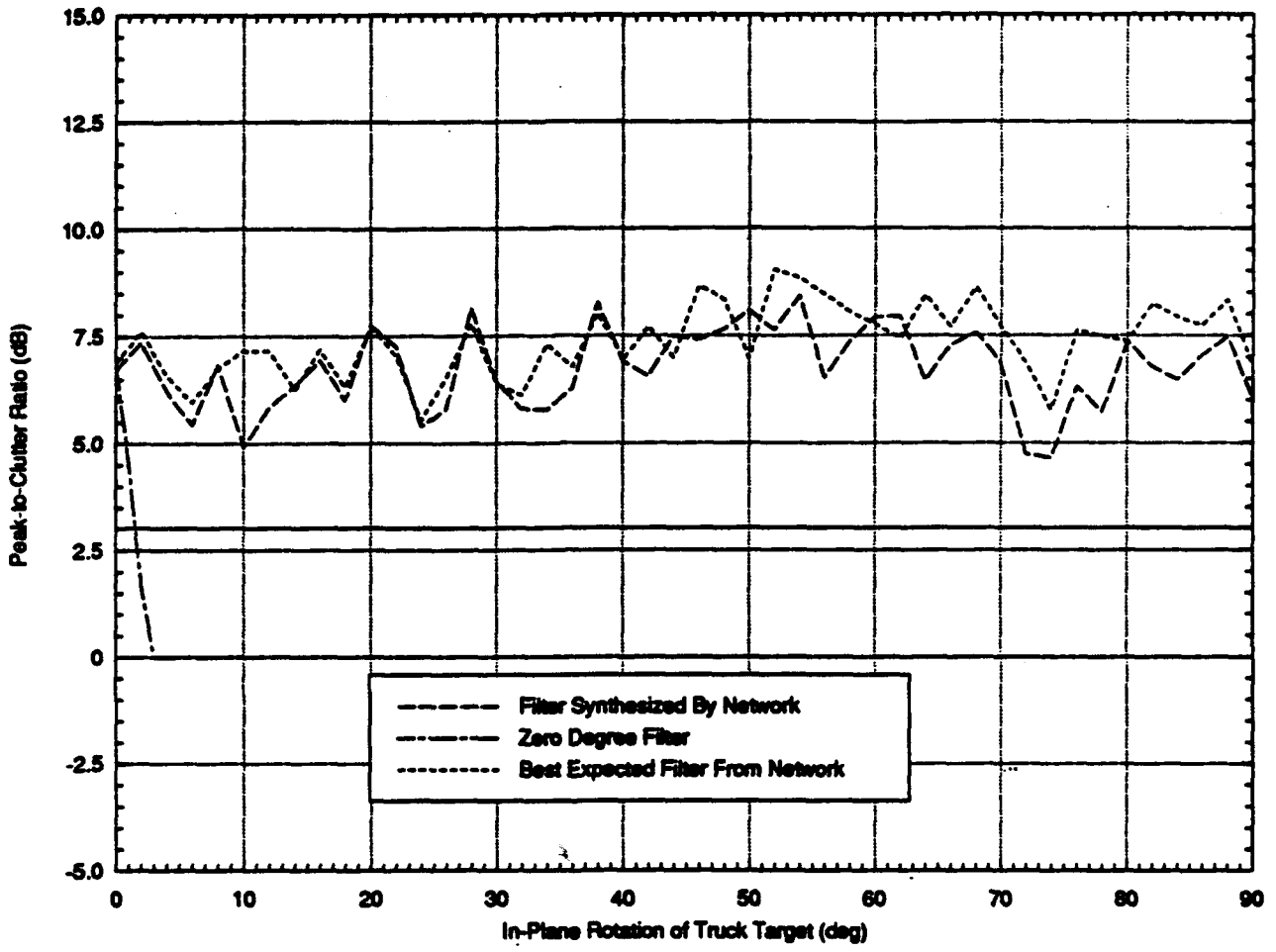


Figure 7.8 Filter synthesis results using a back-propagation neural network for the truck on the trbk70 background at truck rotation angels of 0, 2, ..., 90 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 25 computed values from a target-centered 9 x 9 pixel grid.

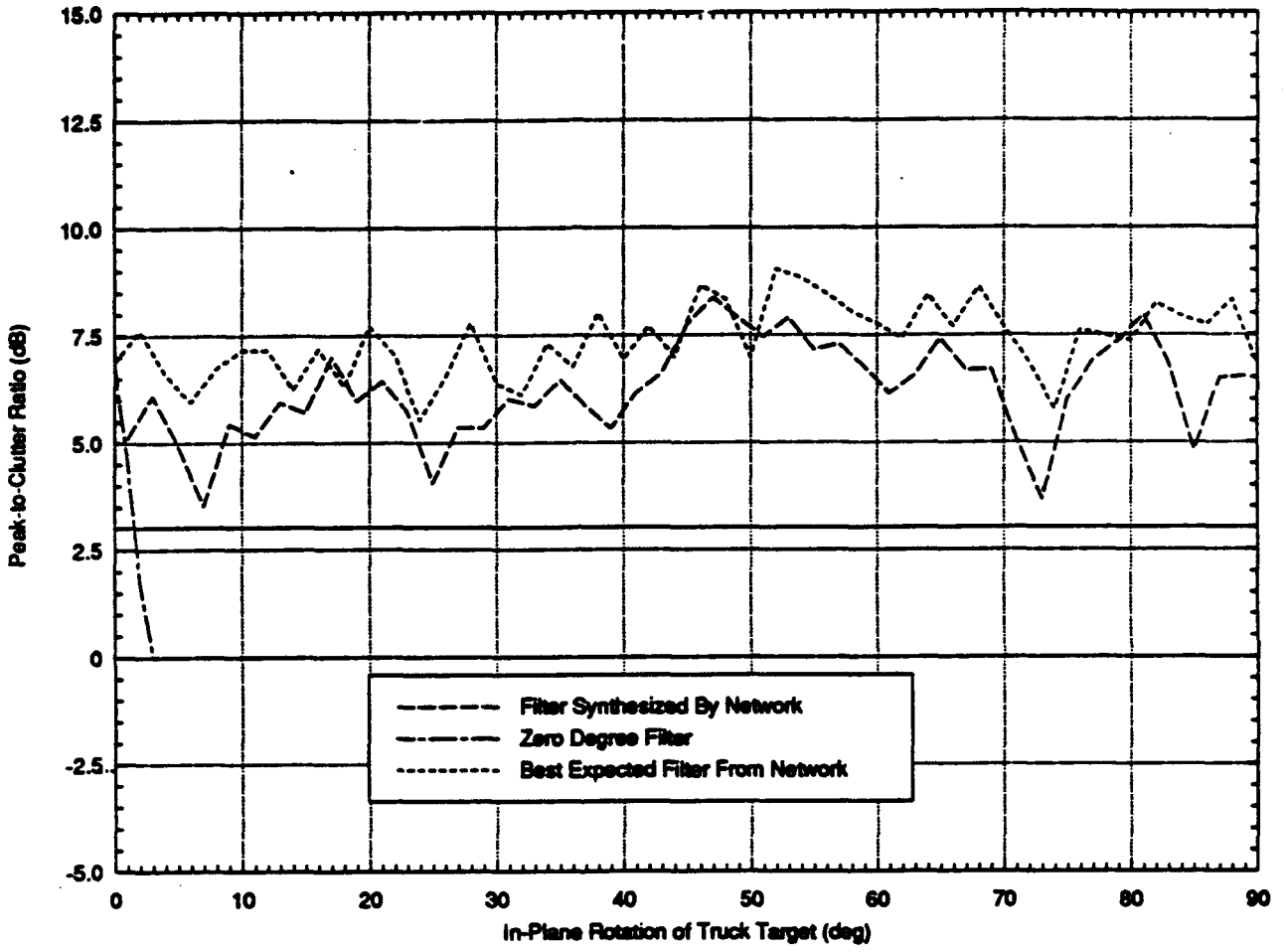


Figure 7.9 Filter synthesis results using a back-propagation neural network for the truck on the trbk70 background at truck rotation angels of 1, 3, ..., 89 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees as inputs using 25 computed values from a target-centered 9 x 9 pixel grid.

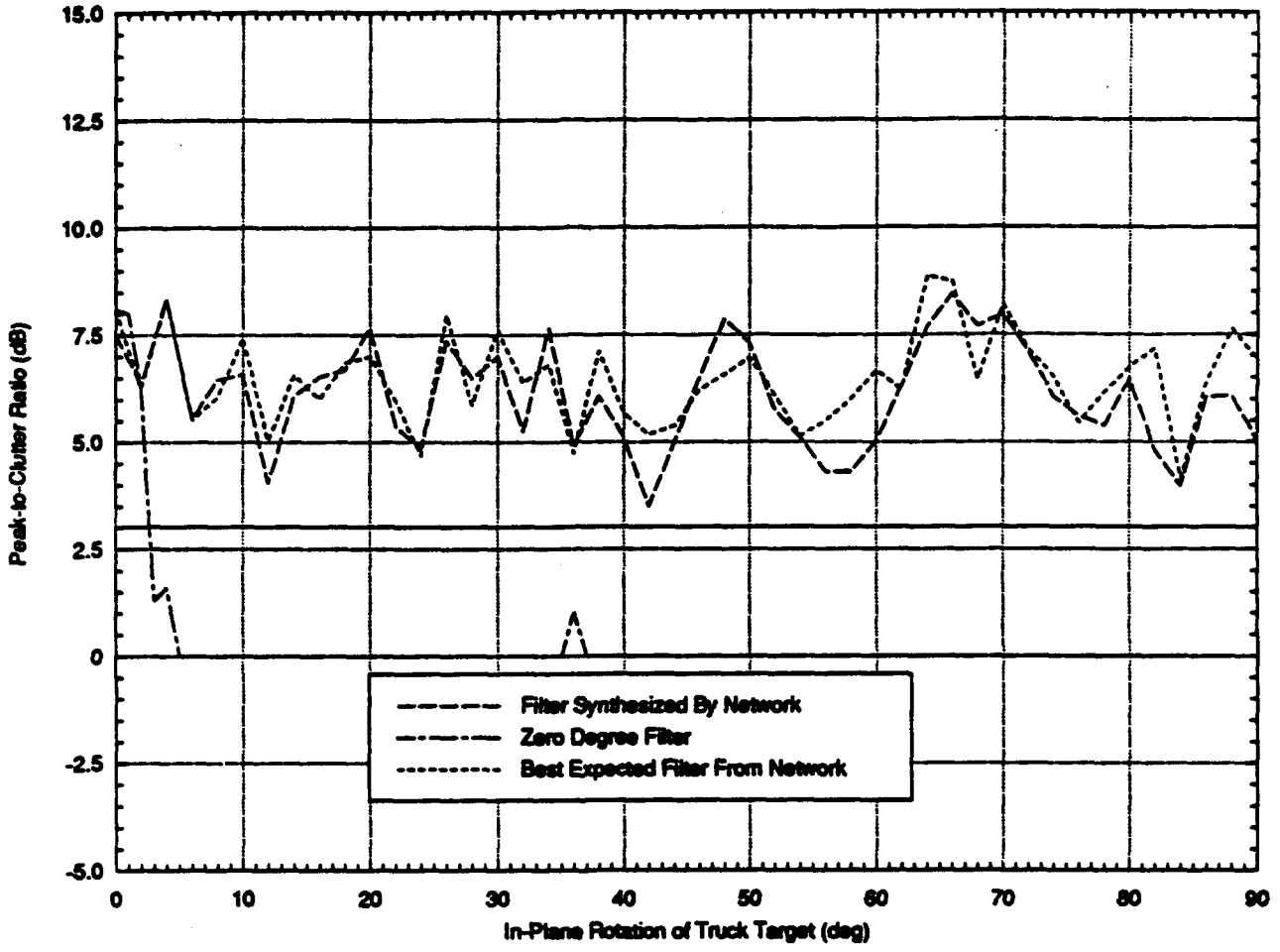


Figure 7.10 Filter synthesis results using a back-propagation neural network for the truck on the newbk80 background at truck rotation angels of 0, 2, ..., 90 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 25 computed values from a target-centered 9 x 9 pixel grid.

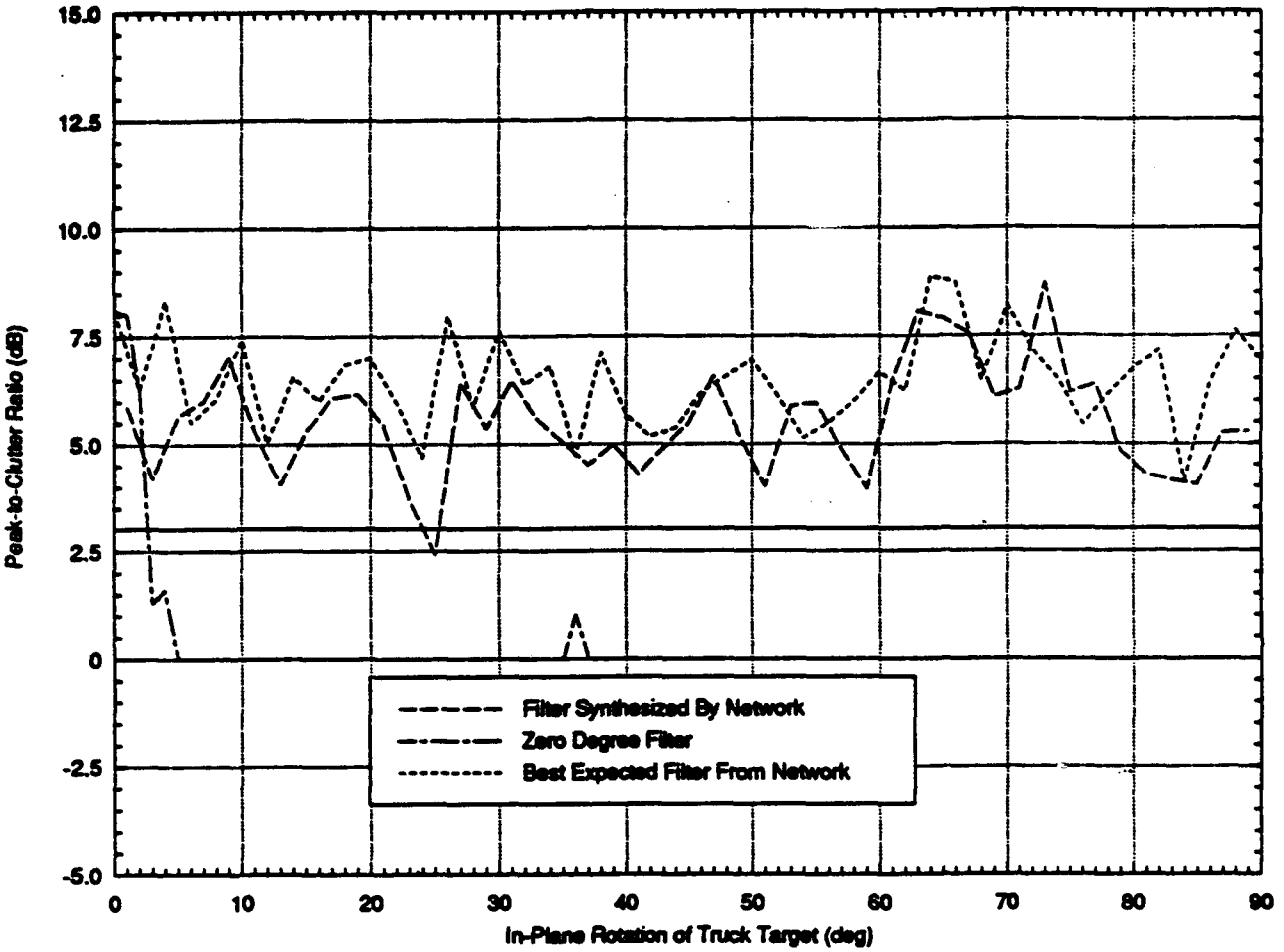


Figure 7.11 Filter synthesis results using a back-propagation neural network for the truck on the newbk80 background at truck rotation angels of 1, 3, ..., 89 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 25 computed values from a target-centered 9 x 9 pixel grid.

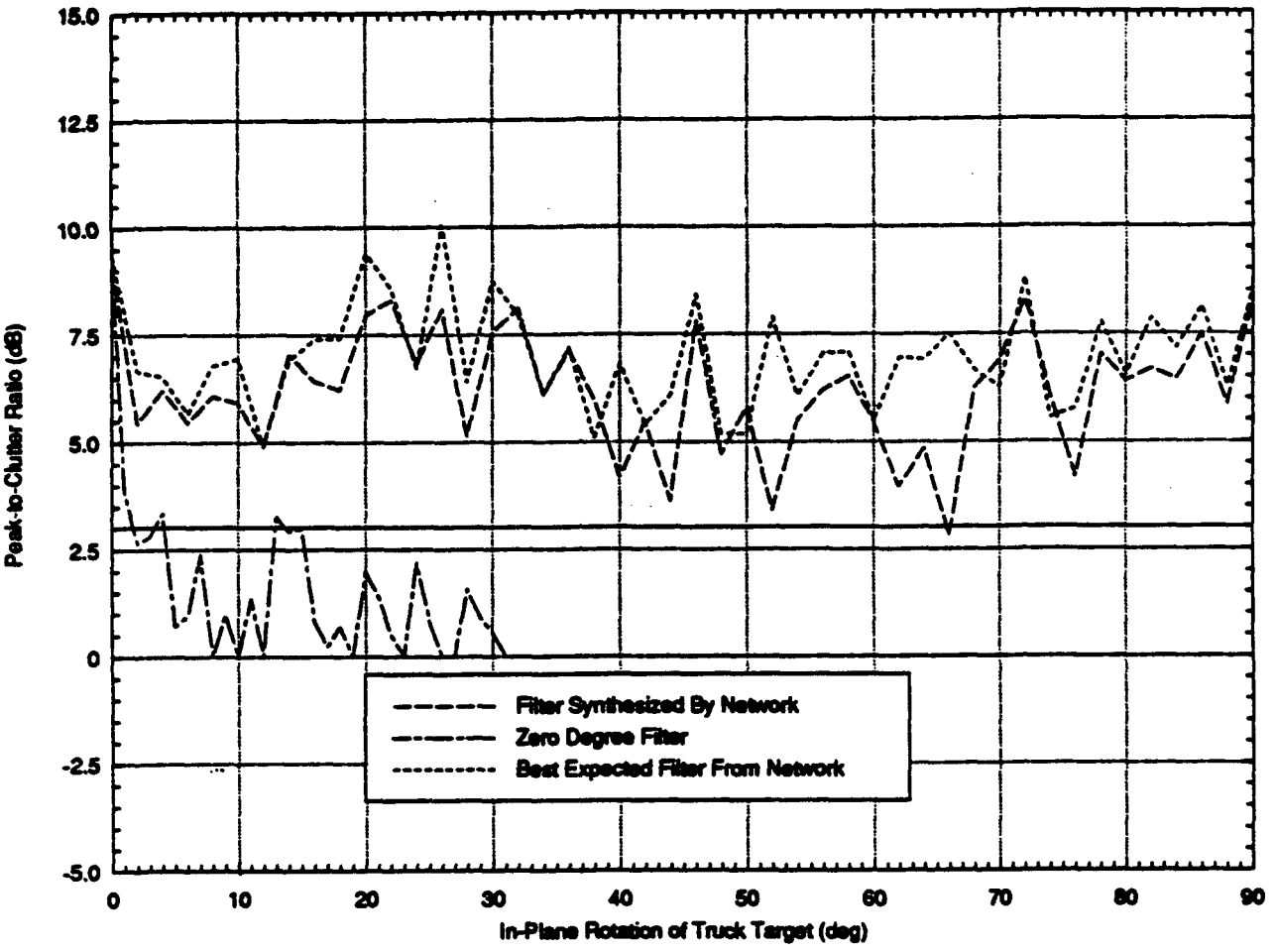


Figure 7.12 Filter synthesis results using a back-propagation neural network for the truck on the bushy background at truck rotation angels of 0, 2, ..., 90 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 25 computed values from a target-centered 9 x 9 pixel grid.

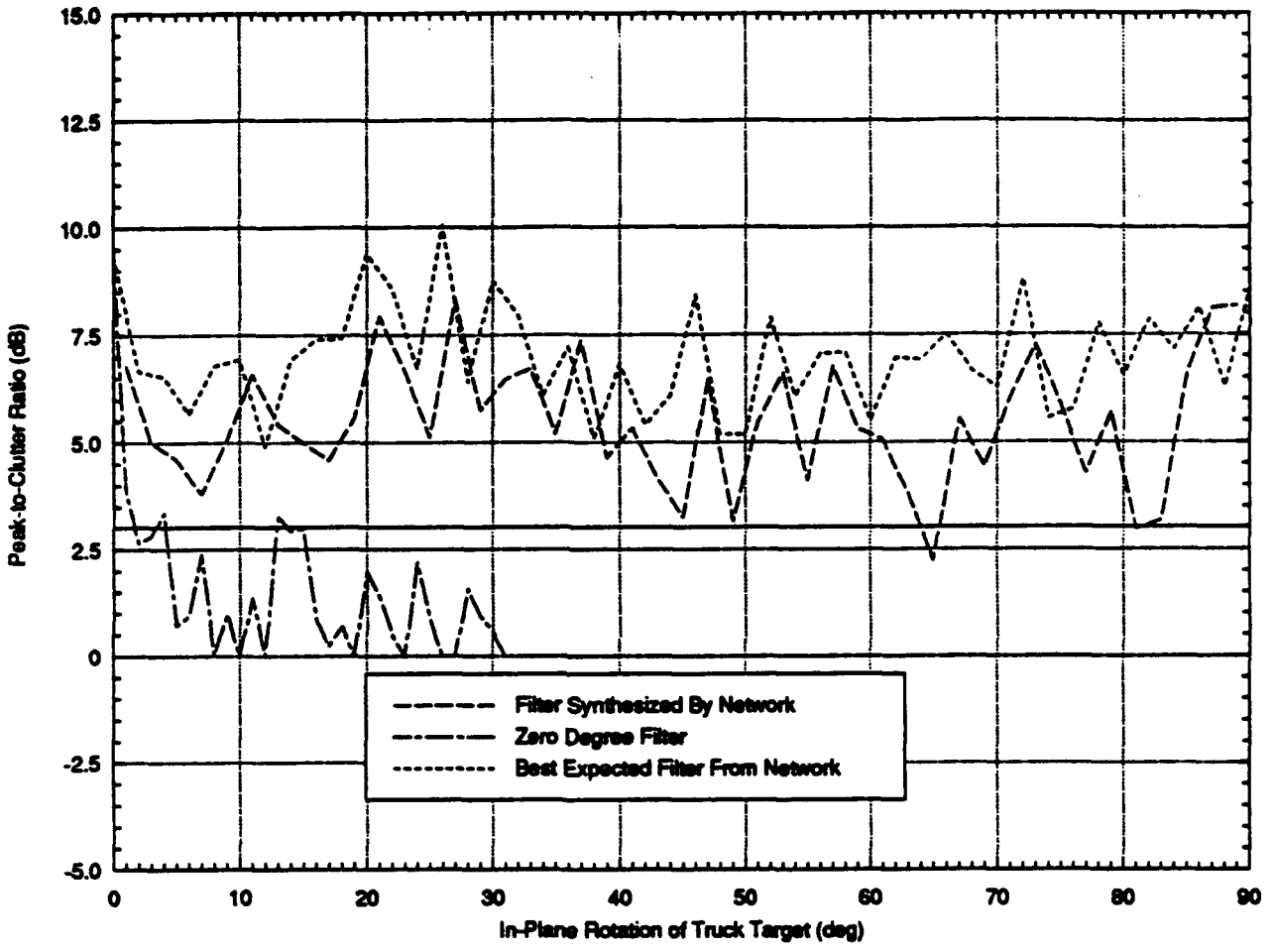


Figure 7.13 Filter synthesis results using a back-propagation neural network for the truck on the bushy background at truck rotation angels of 1, 3, ..., 89 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 25 computed values from a target-centered 9 x 9 pixel grid.

7.4. Back-propagation neural network results (32 wedges, peak-to-clutter)

Using binarized correlator inputs and a TPAF in the filter plane, a back-propagation neural network with 32 inputs (computed from the gray-level input image), 200 hidden neurons, and 600 outputs successfully synthesized filters. The neural network was trained on a set of 138 input scenes of 128 by 128 pixels, as described in Section 7.3.

The back-propagation neural network discussed in Section 7.3 successfully synthesized filters, but to produce filters that perform independently of the input scene background, a different input sampling technique was employed that used 32 values computed from wedges on a target-centered 9 x 9 pixel grid as described in Section 5. Figures 7.14 through 7.21 show graphs of the peak-to-clutter ratio (in dB) versus target rotation (in degrees) for a variety of cases. These results are summarized in Table 7.3. The inputs were 32 values from a target-centered 9 x 9 pixel grid as described in Section 5. The outputs were the 600 coded values used to generate a 10-60 bandpass 3 x 3 superpixel TPAF. The "zero degree filter", "best expected filter", and "filter synthesized by network" curves have the same meaning as in Section 7.2. Performance depends on variations in scene parameters used to test the network and is summarized in Table 7.3.

When tested on the city70 background at angles of 0, 2, ..., 90 degrees (which were used in training) the neural network performance almost matched the best expected performance. When different backgrounds are used at the training angles the network performance degraded, but not as much as for the

results discussed in Section 7.3. As expected, at testing rotation angles (1, 3, ..., 89 degrees) the network performance degradation increased. For the 32-wedge sampling technique the network performed better compared with the sampling technique discussed in Section 7.3 at interpolating the 600 coded filter values for different input scene backgrounds but worse for testing angles. However, the network performance was adequate for pattern recognition (above the 3 dB line) by optical correlation in all but 8 out of 318 testing cases.

Background	Angles of Rotation	Comments
Training-		
City70	0, 2, ..., 90	The performance of the network matches the best expected performance almost exactly.
Testing-		
City70	1, 3, ..., 89	The general performance of the network follows the best expected performance. Performance drops below the 3 dB line for rotation angles of 37 and 83 degrees.
Trbk70	0, 2, ..., 90	The general performance of the network follows the best expected performance almost exactly.
Trbk70	1, 3, ..., 89	The general performance of the network follows the best expected performance almost exactly.
Newbk80	0, 2, ..., 90	The general performance of the network follows the best expected performance. Performance drops below the 3 dB line for a rotation angle of 42 degrees.
Newbk80	1, 3, ..., 89	The general performance of the network follows the best expected performance. Performance drops below the 3 dB line for rotation angles of 41 and 45 degrees.
Bushy	0, 2, ..., 90	The general performance of the network follows the best expected performance.
Bushy	1, 3, ..., 89	The general performance of the network follows the best expected performance. Performance drops below the 3 dB line for rotation angles of 45, 55, and 83 degrees.

Table 7.3 Shows the performance of a filter synthesized by a back-propagation neural network using the 32 wedge sampling technique and the peak-to-clutter metric for a truck rotated on a variety of backgrounds.

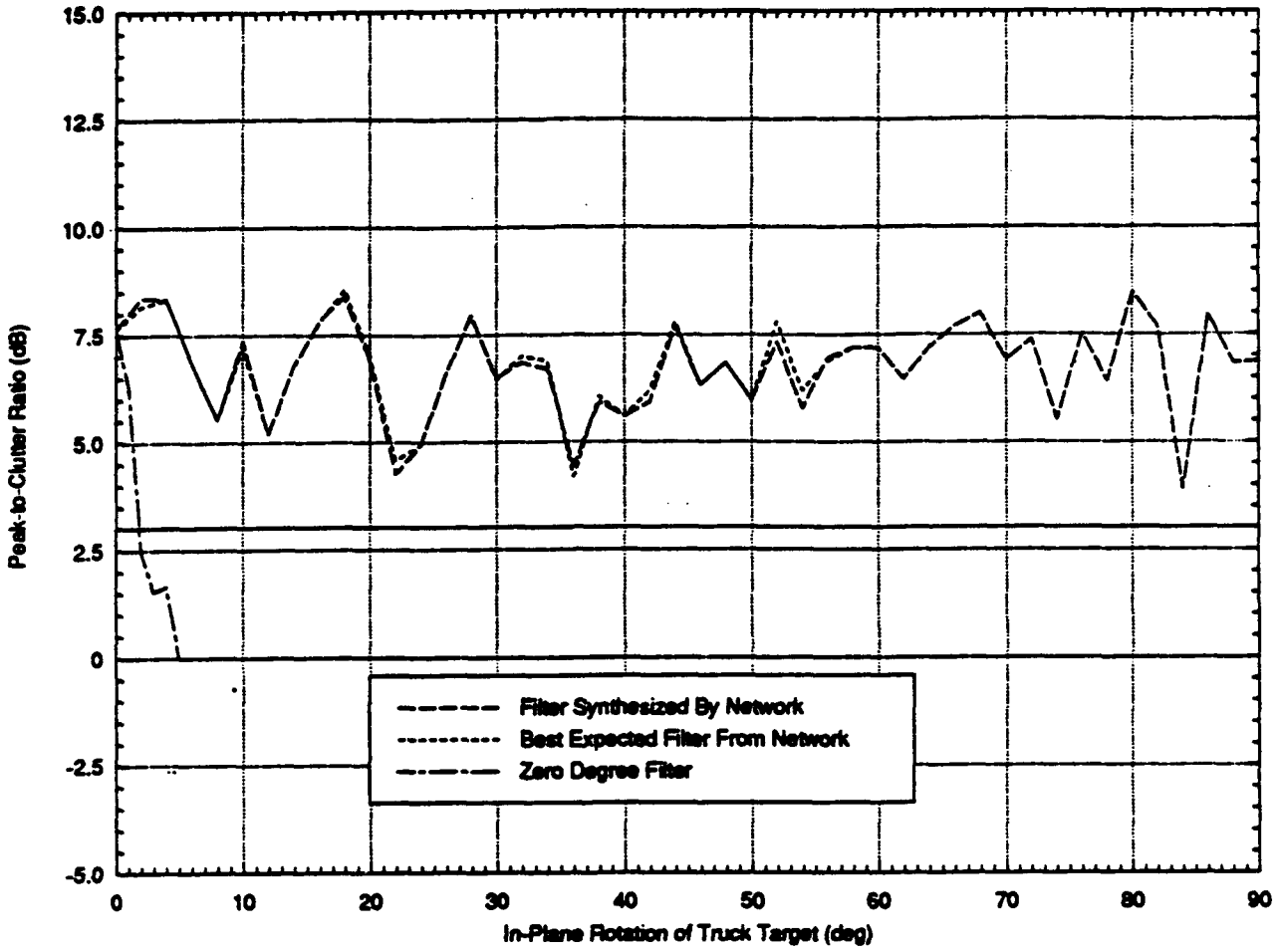


Figure 7.14 Filter synthesis results using a back-propagation neural network for the truck on the city70 background at truck rotation angles of 0, 2, ..., 90 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 31 computed values from a target-centered 9 x 9 pixel grid.

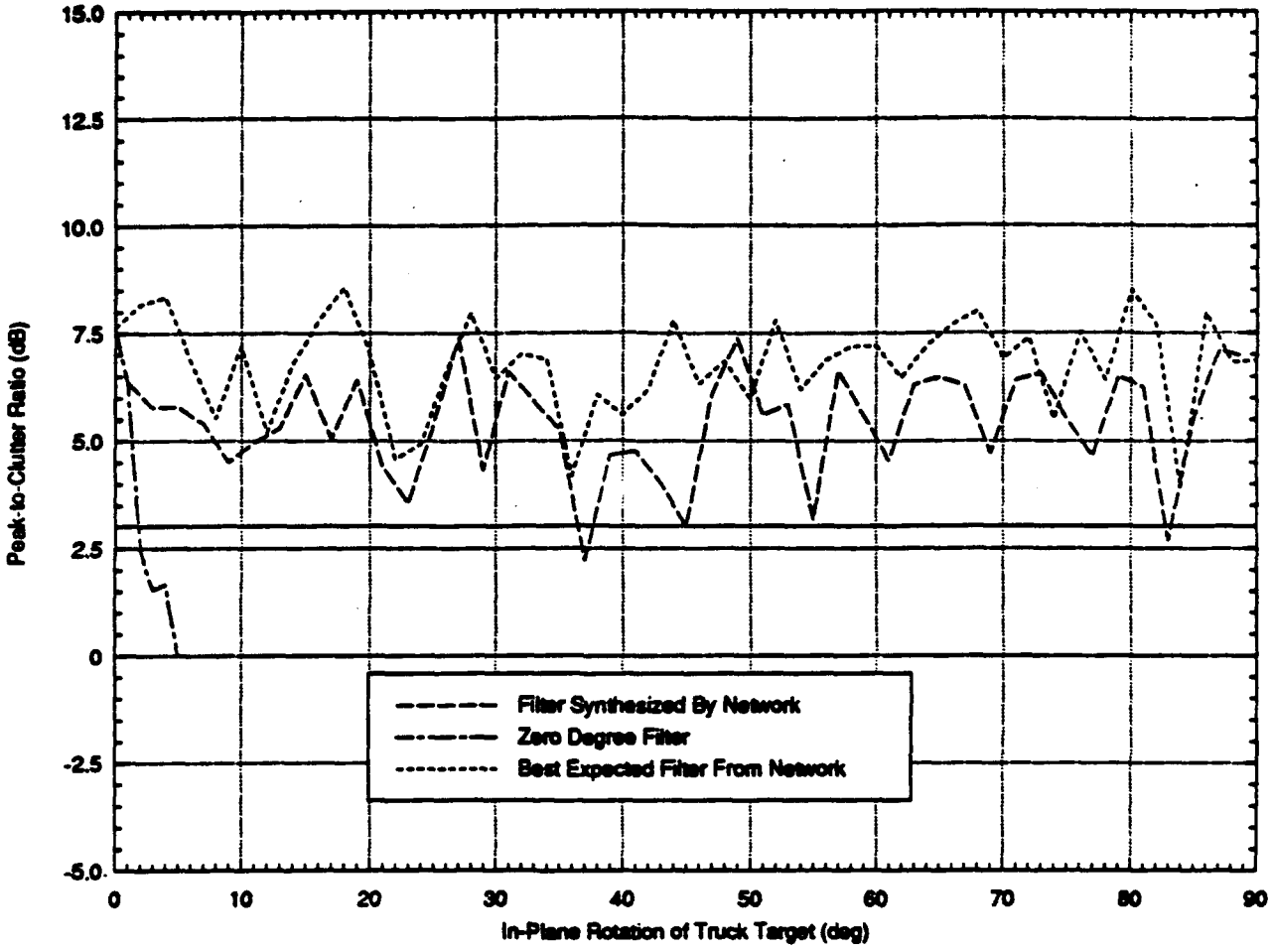


Figure 7.15 Filter synthesis results using a back-propagation neural network for the truck on the city70 background at truck rotation angles of 1, 3, ..., 89 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 31 computed values from a target-centered 9 x 9 pixel grid.

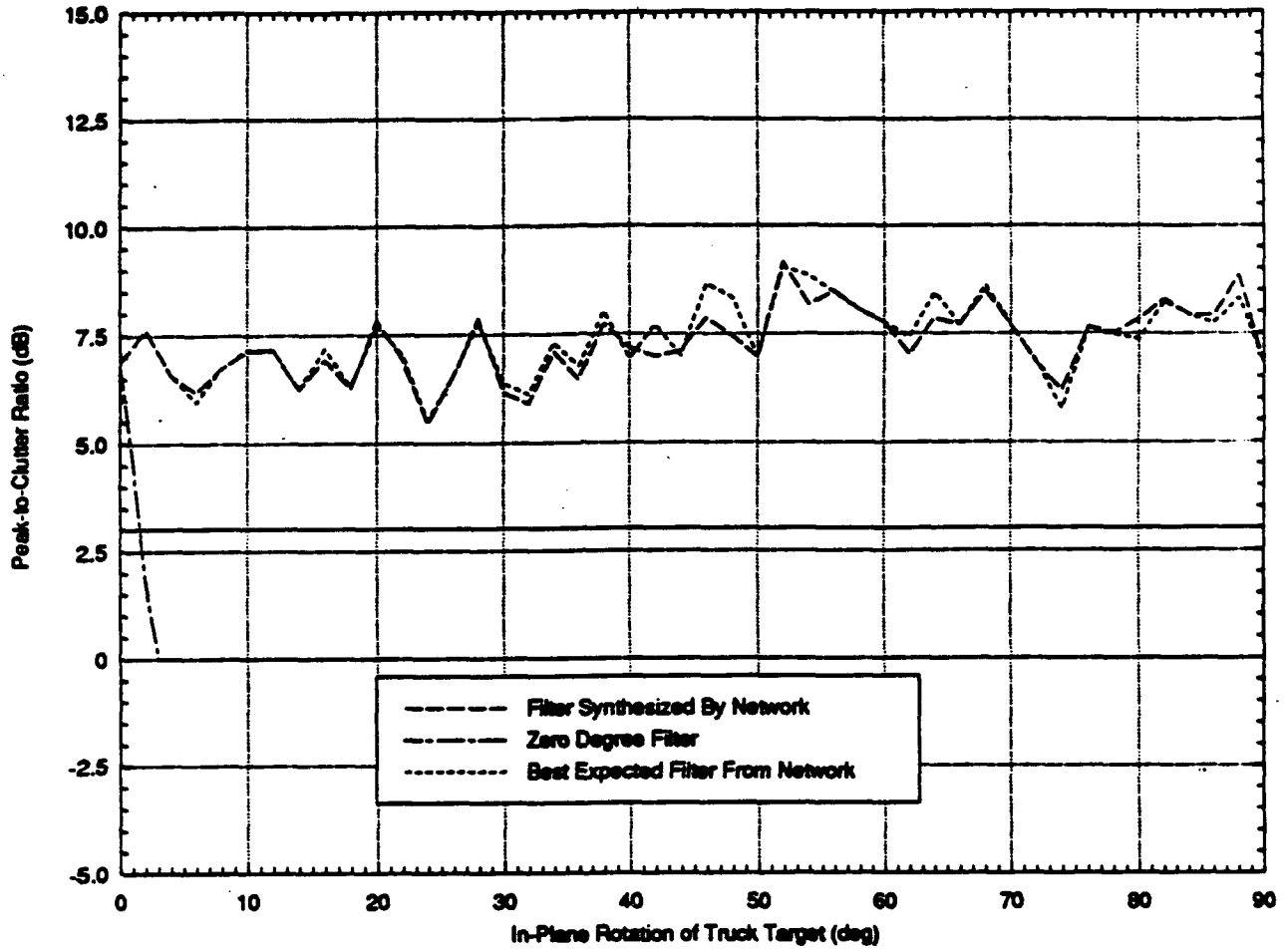


Figure 7.16 Filter synthesis results using a back-propagation neural network for the truck on the trbk70 background at truck rotation angles of 0, 2, ..., 90 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 31 computed values from a target-centered 9 x 9 pixel grid.

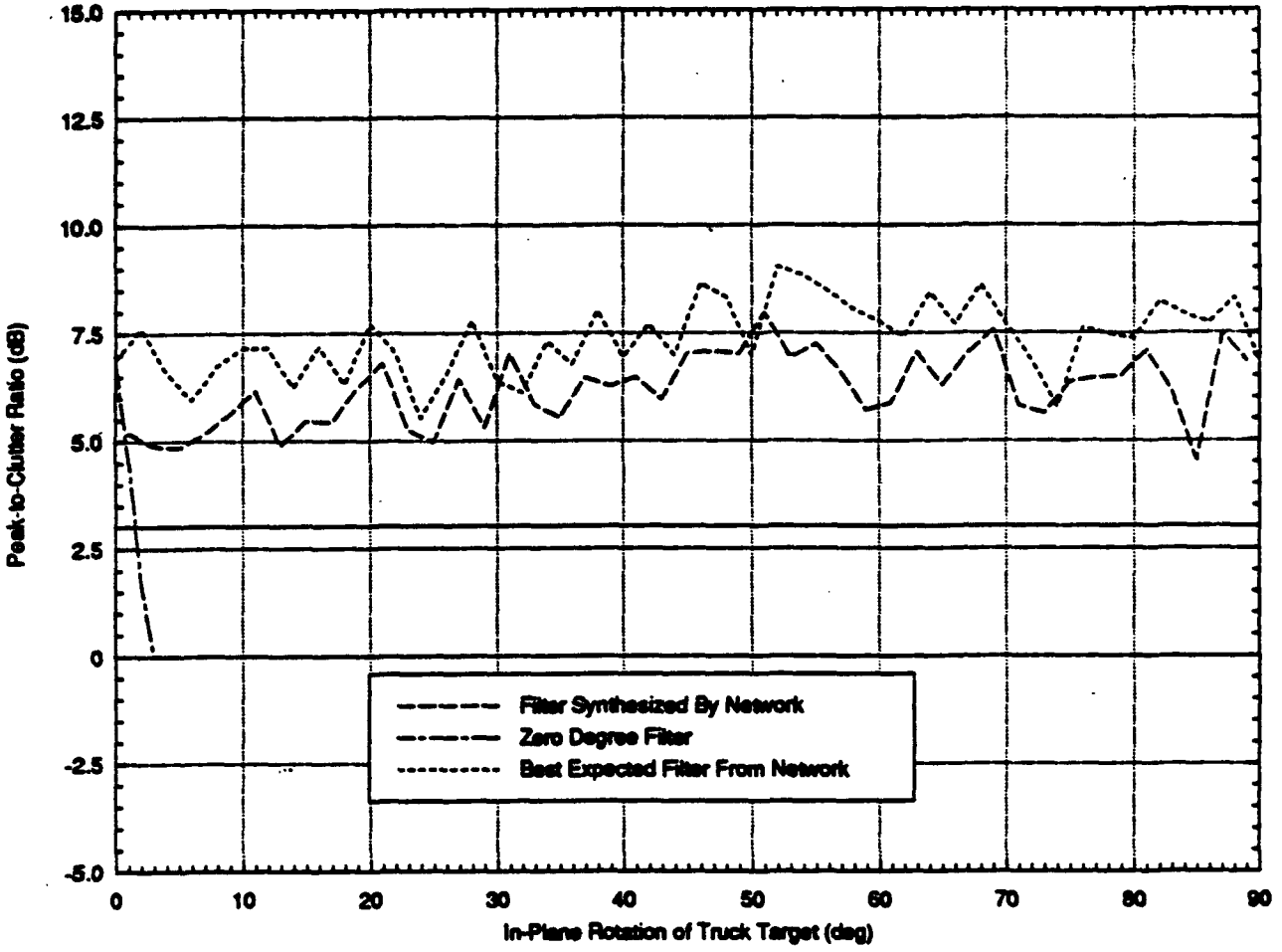


Figure 7.17 Filter synthesis results using a back-propagation neural network for the truck on the trbk70 background at truck rotation angles of 1, 3, ..., 89 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 31 computed values from a target-centered 9 x 9 pixel grid.

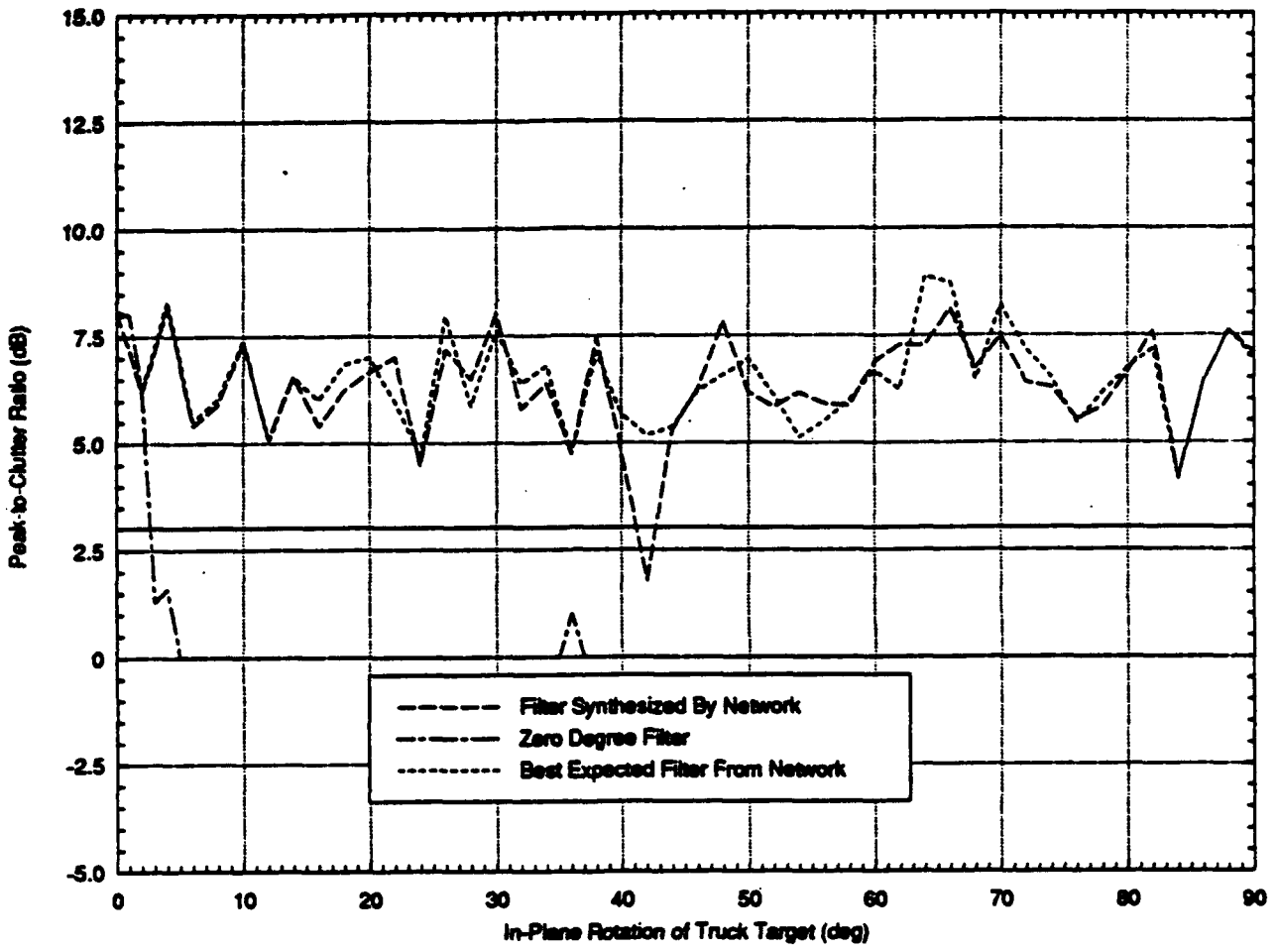


Figure 7.18 Filter synthesis results using a back-propagation neural network for the truck on the newbk80 background at truck rotation angles of 0, 2, ..., 90 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 31 computed values from a target-centered 9 x 9 pixel grid.

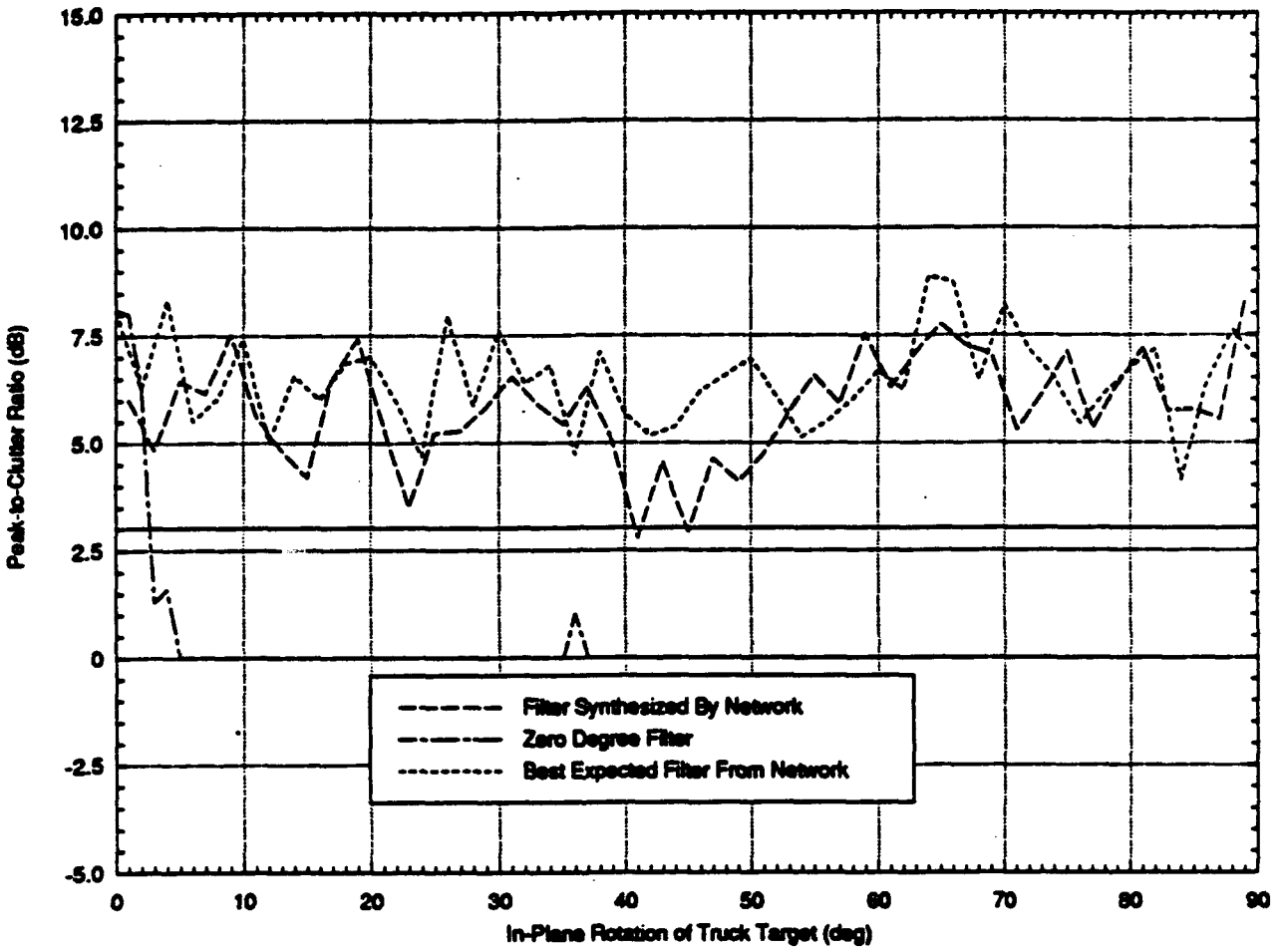


Figure 7.19 Filter synthesis results using a back-propagation neural network for the truck on the newbk80 background at truck rotation angles of 1, 3, ..., 89 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 31 computed values from a target-centered 9 x 9 pixel grid.

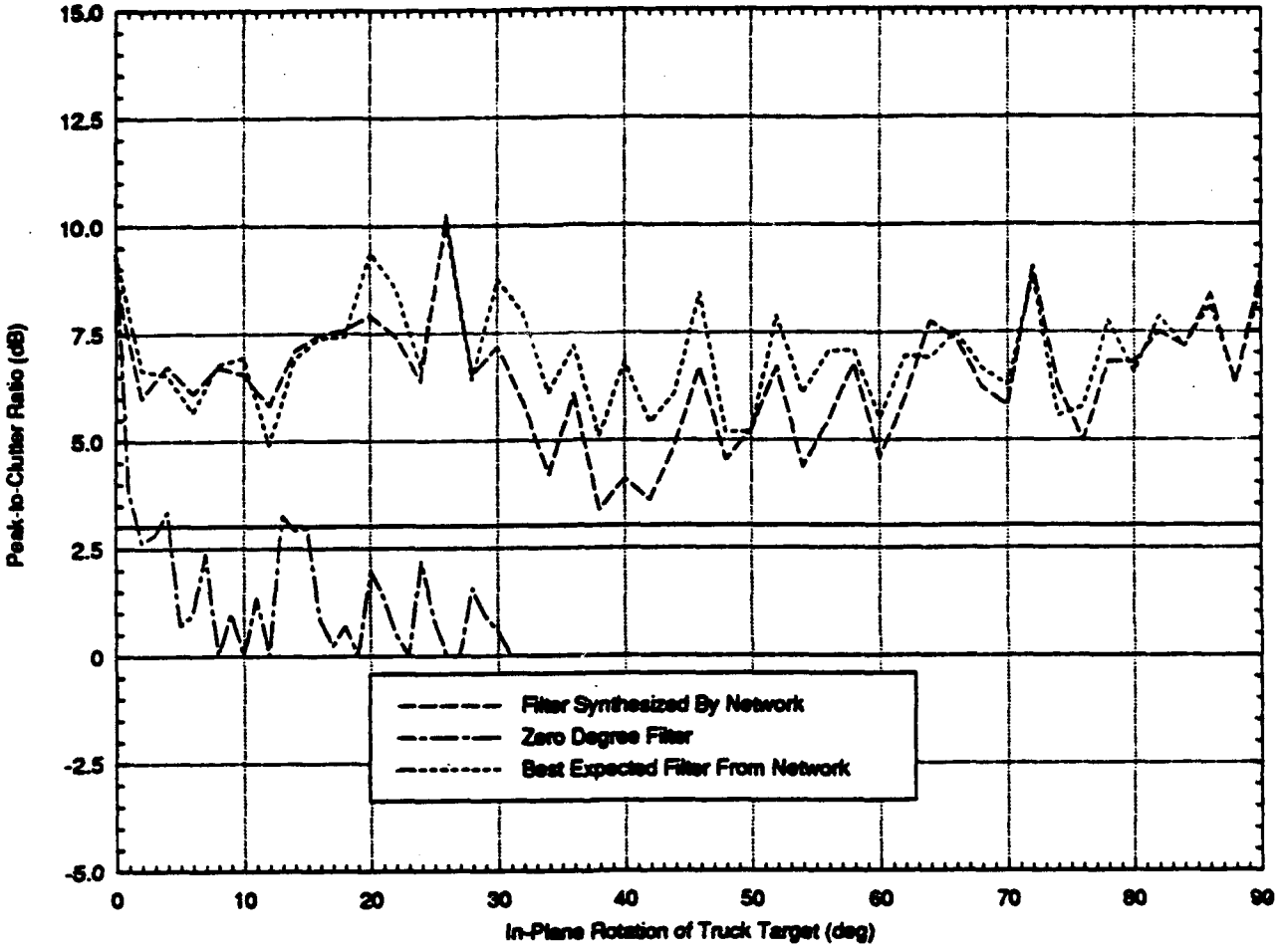


Figure 7.20 Filter synthesis results using a back-propagation neural network for the truck on the bushy background at truck rotation angles of 0, 2, ..., 90 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 31 computed values from a target-centered 9 x 9 pixel grid.

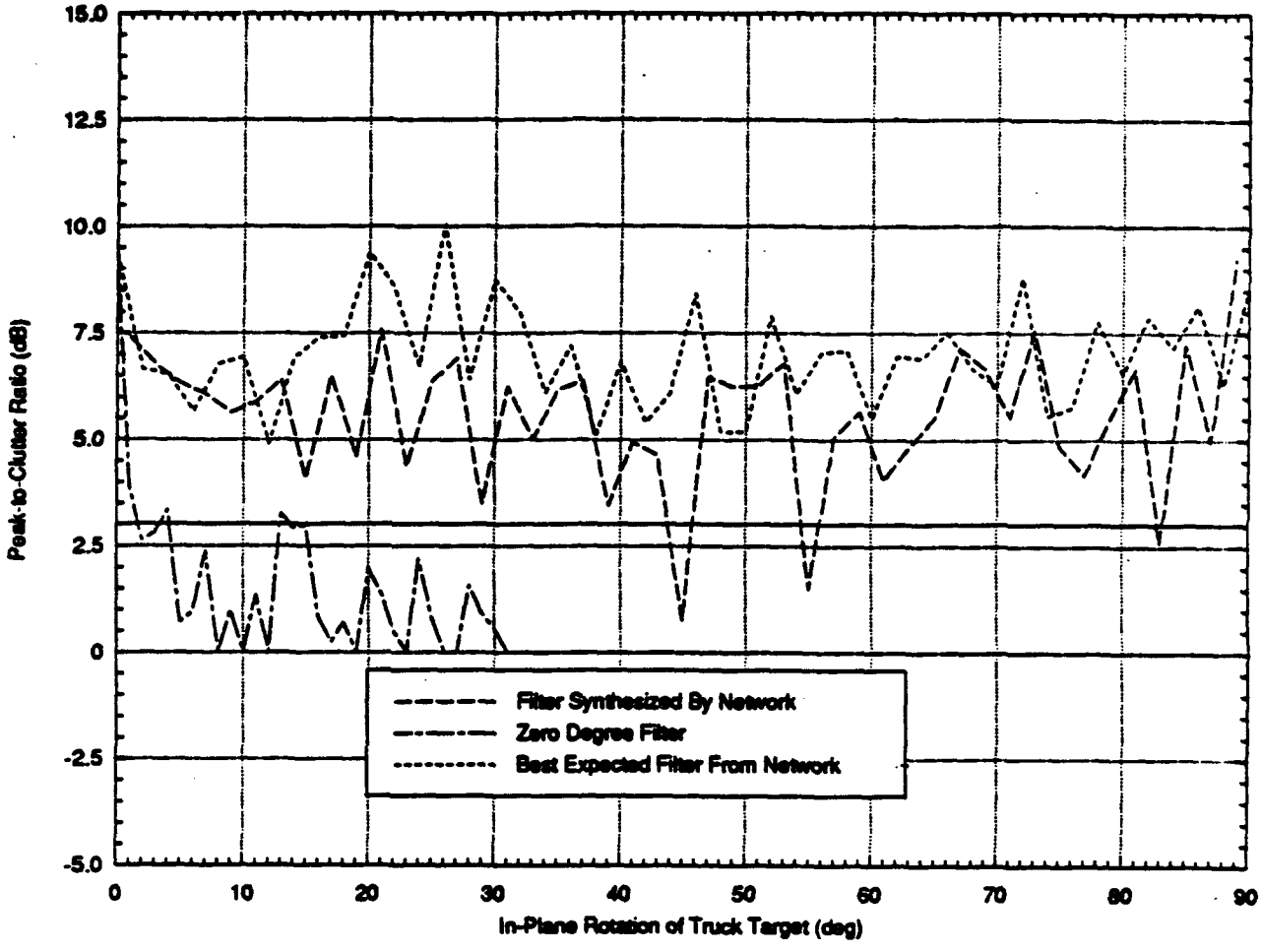


Figure 7.21 Filter synthesis results using a back-propagation neural network for the truck on the bushy background at truck rotation angles of 1, 3, ..., 89 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 31 computed values from a target-centered 9 x 9 pixel grid.

7.5. Stretch and hammer neural network results (32 wedges, peak-to-clutter)

Using binarized correlator inputs and a TPAF in the filter plane, a stretch and hammer neural network with 32 inputs (computed from the gray-level input image) and 600 outputs successfully synthesized filters. The neural network was trained on a set of 138 input scenes of 128 by 128 pixels as described in Section 7.3.

The results discussed in Section 7.4 indicate performance independent of the background, however, there was a slight degradation in performance at angles not included in training. This effect is expected since the wedge sampling technique is sensitive to rotation angles. For comparison, a stretch and hammer neural network was trained using the same wedge inputs. Figures 7.22 through 7.29 show graphs of the peak-to-clutter ratio (in dB) versus target rotation (in degrees) for a variety of cases. These results are summarized in Table 7.4. The inputs were 32 values from a target-centered 9 x 9 pixel grid as described in Section 5. The outputs were the 600 coded values used to generate a 10-60 bandpass 3 x 3 superpixel TPAF. The "zero degree filter", "best expected filter", and "filter synthesized by network" curves have the same meaning as in Section 7.2. Performance depends on variations in scene parameters used to test the network and therefore is summarized in Table 7.4.

When tested on the city70 background at angles of 0, 2, ..., 90 degrees (which were used in training) the neural network performance, by definition, matched the best expected performance exactly. At training angles the network

performance degraded slightly for the trbk70 background and more so for the newbk80 and bushy backgrounds. At testing rotation angles (1, 3, ..., 89 degrees) the neural network performance was adequate for pattern recognition (above the 3 dB line) by optical correlation in all but 42 out of 318 testing cases.

Background	Angles of Rotation	Comments
Training-		
City70	0, 2, ..., 90	The performance of the network matches the best expected performance exactly.
Testing-		
City70	1, 3, ..., 89	The performance of the network drops below the 3 dB line in 8 out of 45 tests, filters were successfully synthesized approximately 80% of the time.
Trbk70	0, 2, ..., 90	The general performance of the network follows the best expected performance.
Trbk70	1, 3, ..., 89	The general performance of the network is adequate for pattern recognition. Performance drops below the 3 dB line for rotation angles of 43 and 75 degrees.
Newbk80	0, 2, ..., 90	The general performance of the network is adequate for pattern recognition. Performance drops below the 3 dB line for rotation angles of 24, 40, 42, and 46 degrees.
Newbk80	1, 3, ..., 89	The performance of the network drops below the 3 dB line in 8 out of 45 tests, filters were successfully synthesized approximately 80% of the time.
Bushy	0, 2, ..., 90	The performance of the network drops below the 3 dB line in 8 out of 46 tests, filters were successfully synthesized approximately 80% of the time.
Bushy	1, 3, ..., 89	The performance of the network drops below the 3 dB line in 13 out of 45 tests, filters were successfully synthesized approximately 65% of the time.

Table 7.4 Shows the performance of a filter synthesized by a stretch and hammer neural network using the 32 wedge sampling technique and the peak-to-clutter metric for a truck on a variety of backgrounds.

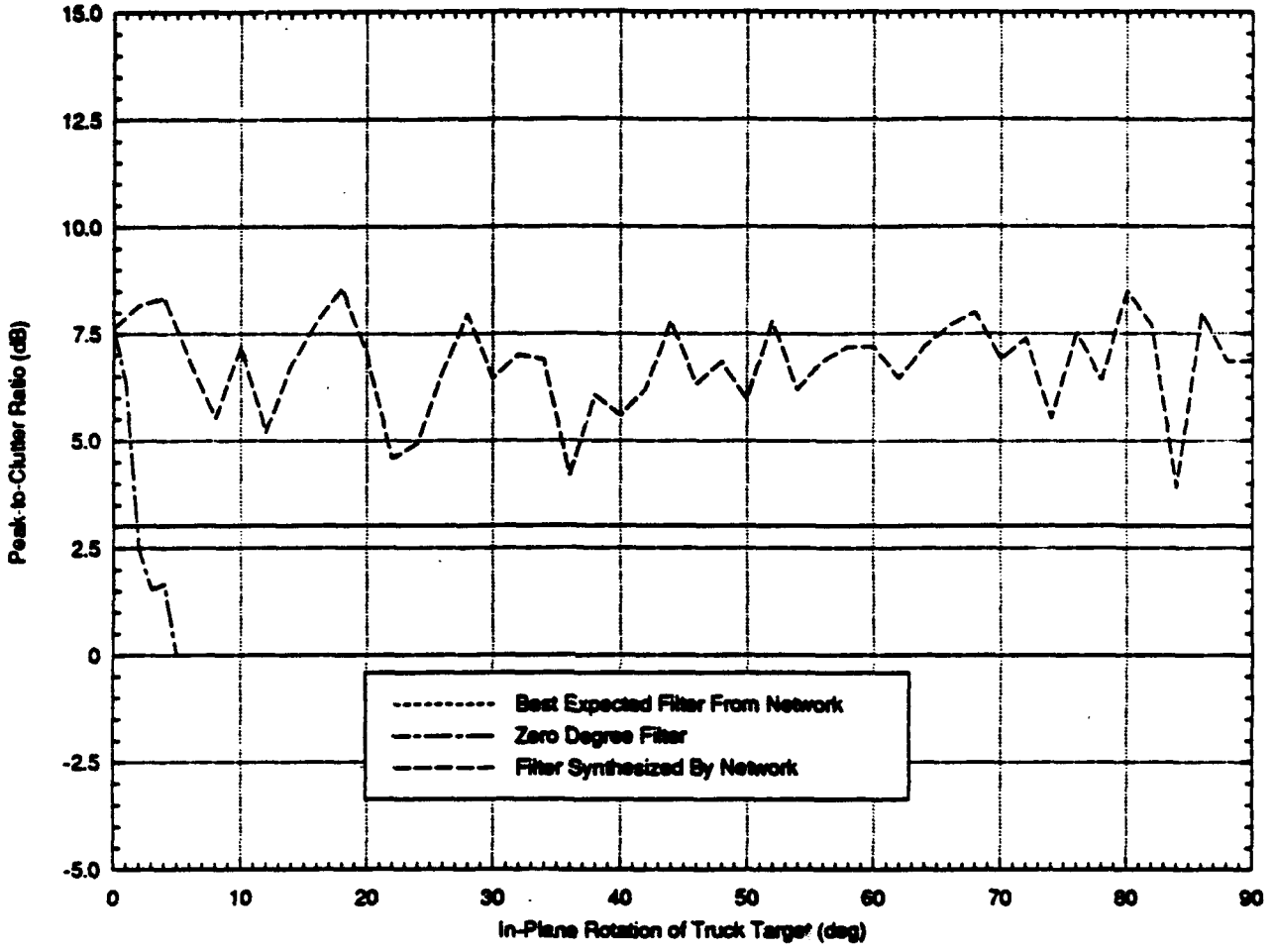


Figure 7.22 Filter synthesis results using a stretch and hammer neural network for the truck on the city70 background at truck rotation angles of 0, 2, ..., 90 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 31 computed values from a target-centered 9 x 9 pixel grid.

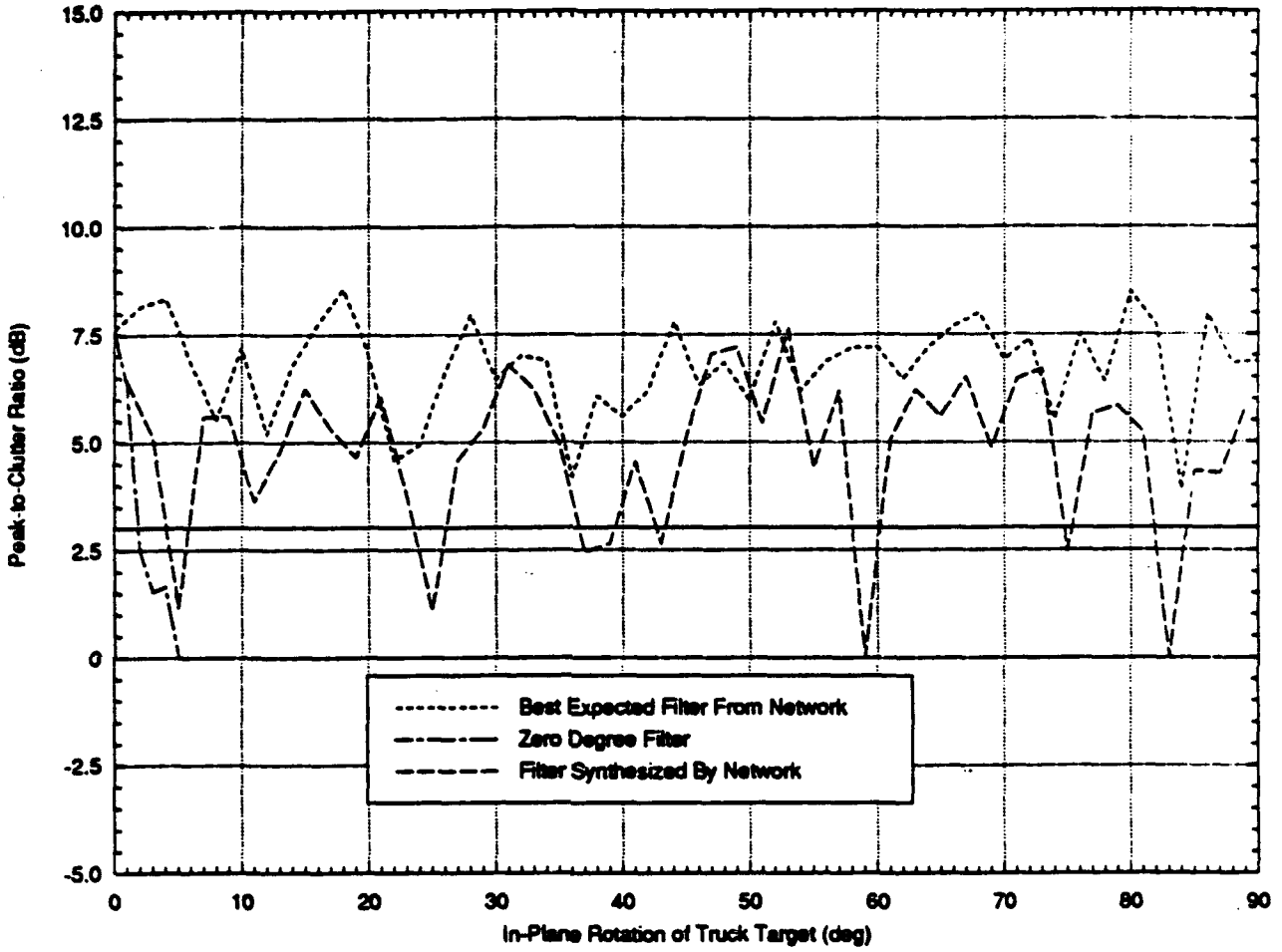


Figure 7.23 Filter synthesis results using a stretch and hammer neural network for the truck on the city70 background at truck rotation angles of 1, 3, ..., 89 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 31 computed values from a target-centered 9 x 9 pixel grid.

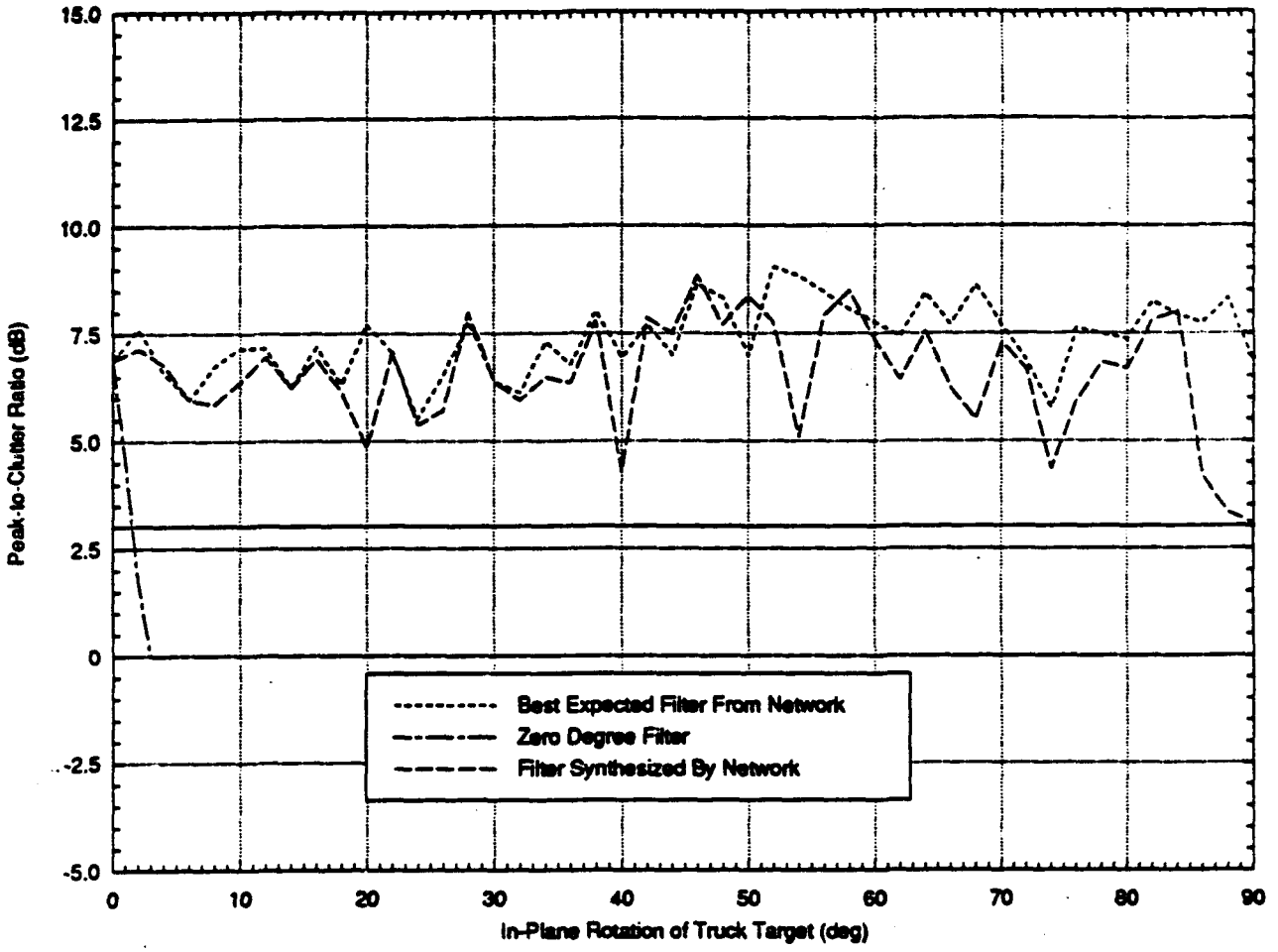


Figure 7.24 Filter synthesis results using a stretch and hammer neural network for the truck on the trbk70 background at truck rotation angles of 0, 2, ..., 90 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 31 computed values from a target-centered 9 x 9 pixel grid.

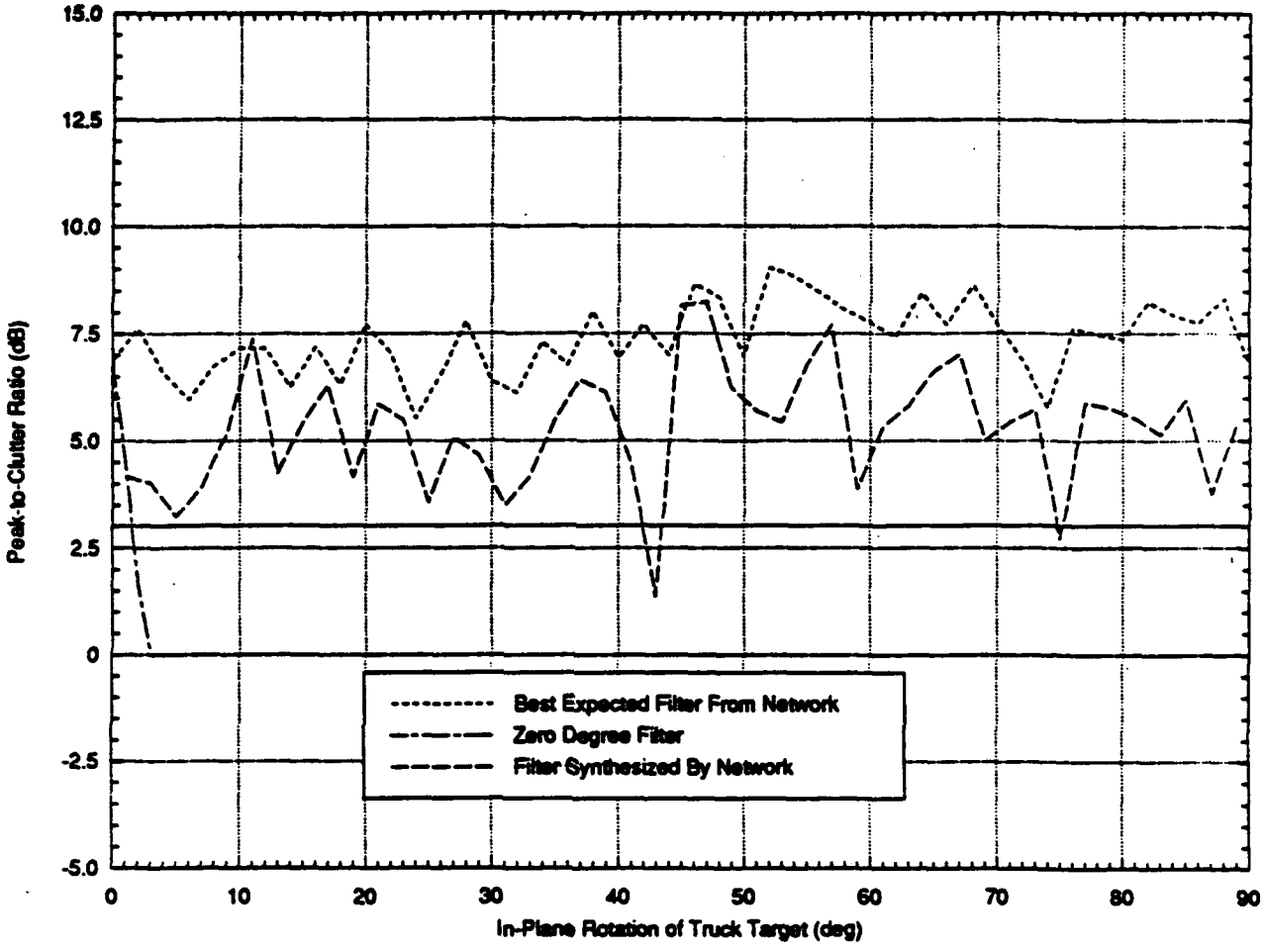


Figure 7.25 Filter synthesis results using a stretch and hammer neural network for the truck on the trbk70 background at truck rotation angles of 1, 3, ..., 89 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 31 computed values from a target-centered 9 x 9 pixel grid.

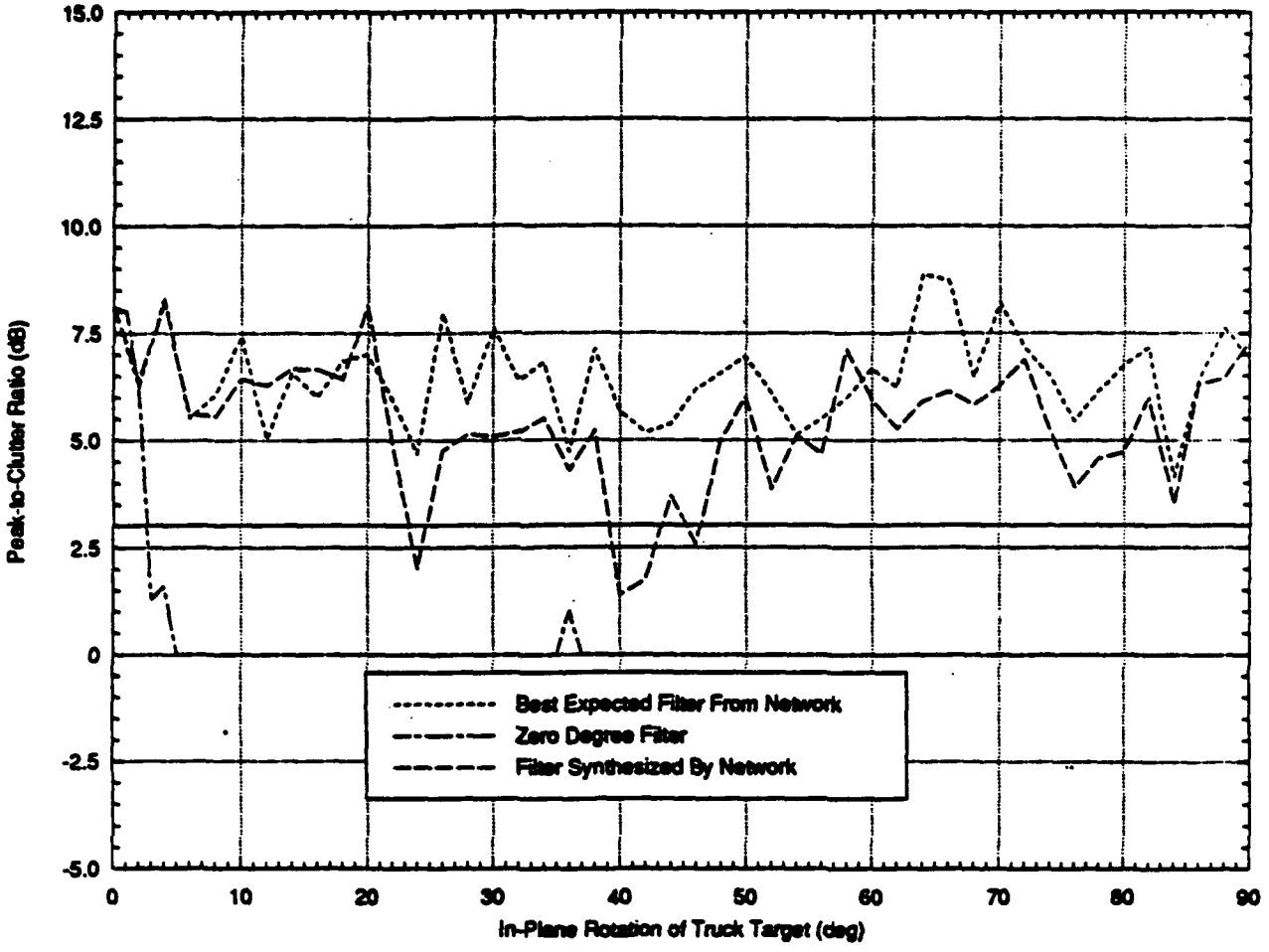


Figure 7.26 Filter synthesis results using a stretch and hammer neural network for the truck on the newbk80 background at truck rotation angles of 0, 2, ..., 90 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 31 computed values from a target-centered 9 x 9 pixel grid.

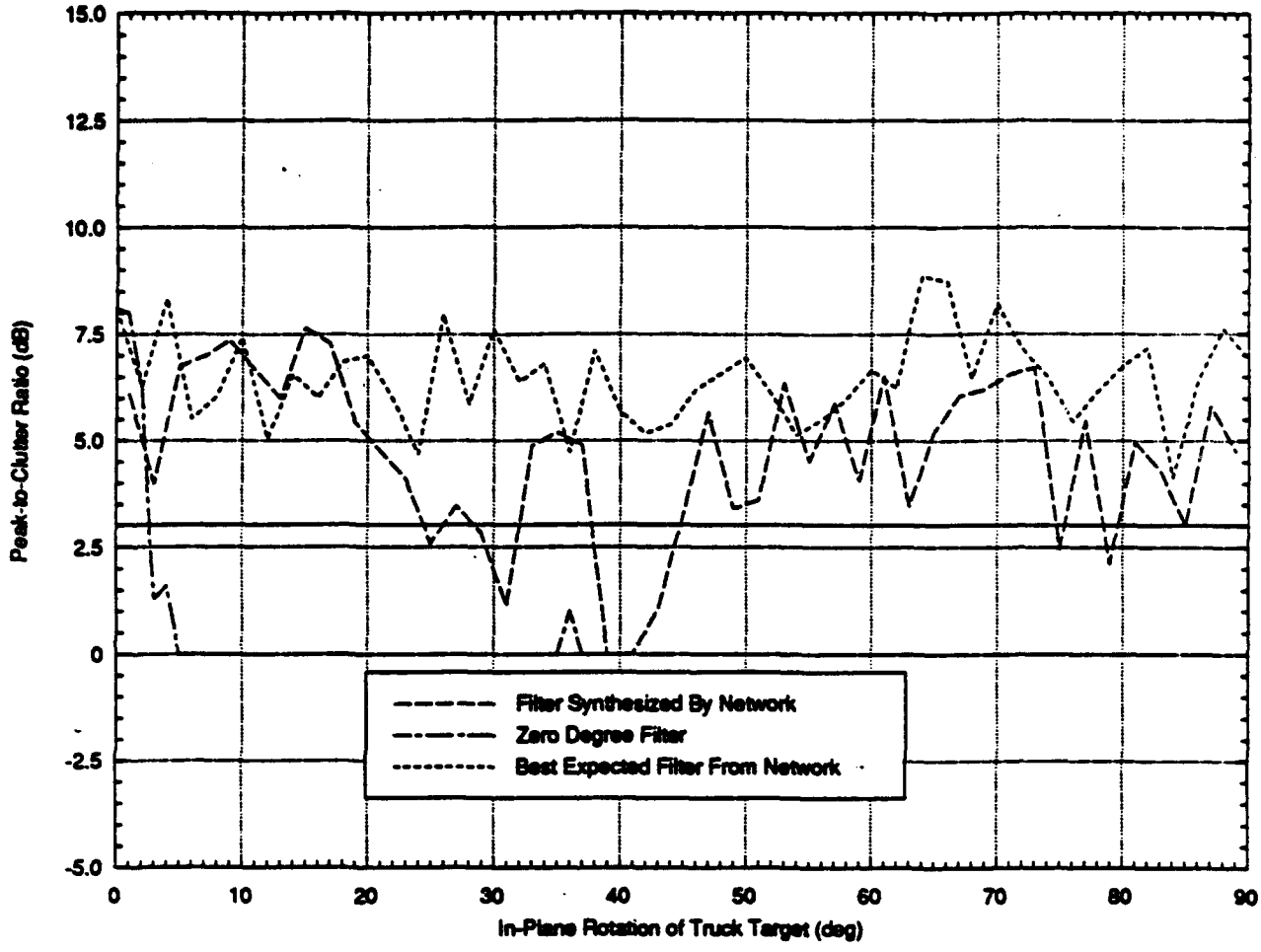


Figure 7.27 Filter synthesis results using a stretch and hammer neural network for the truck on the newbk80 background at truck rotation angles of 1, 3, ..., 89 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 31 computed values from a target-centered 9 x 9 pixel grid.

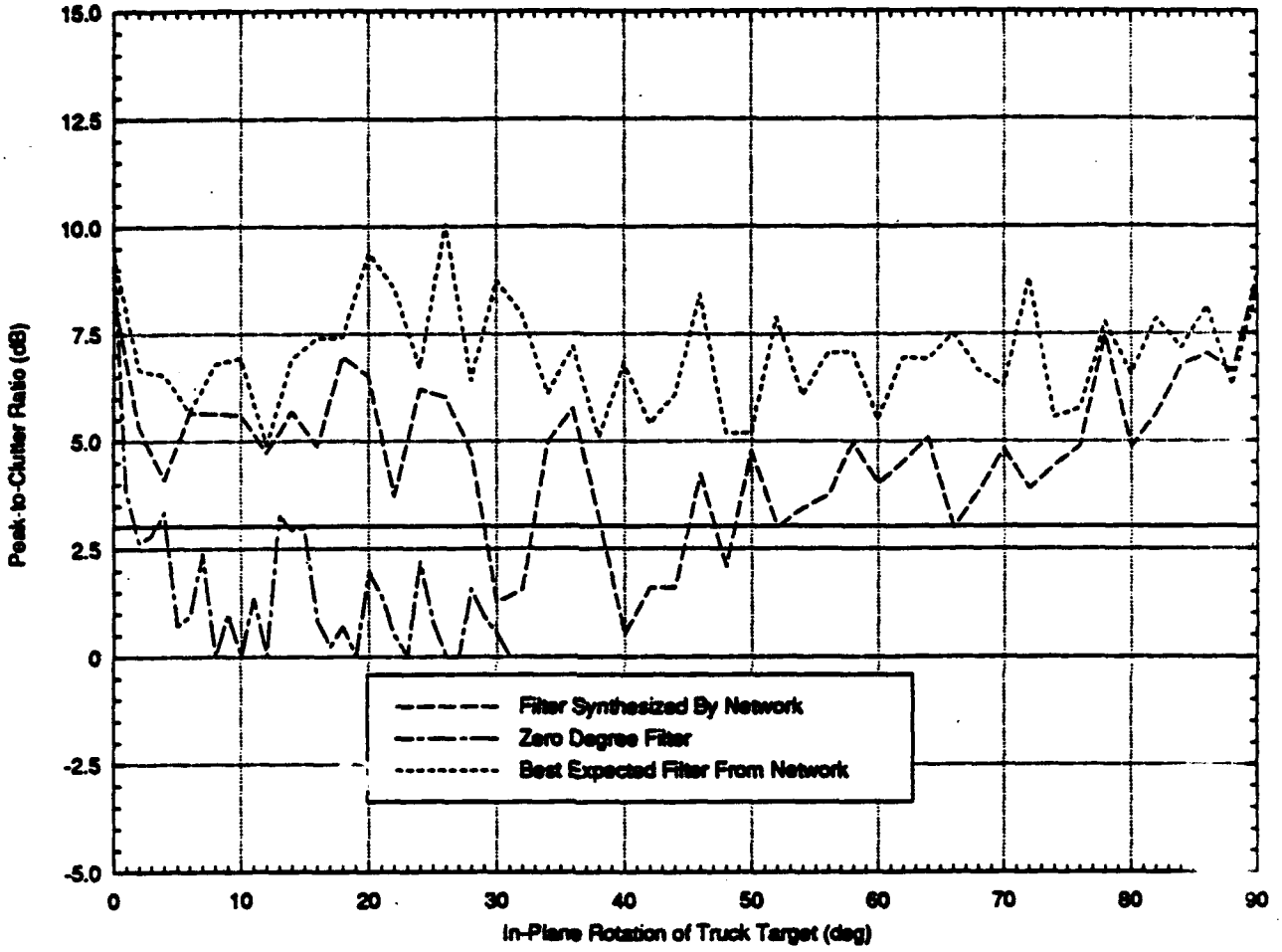


Figure 7.28 Filter synthesis results using a stretch and hammer neural network for the truck on the bushy background at truck rotation angles of 0, 2, ..., 90 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 31 computed values from a target-centered 9 x 9 pixel grid.

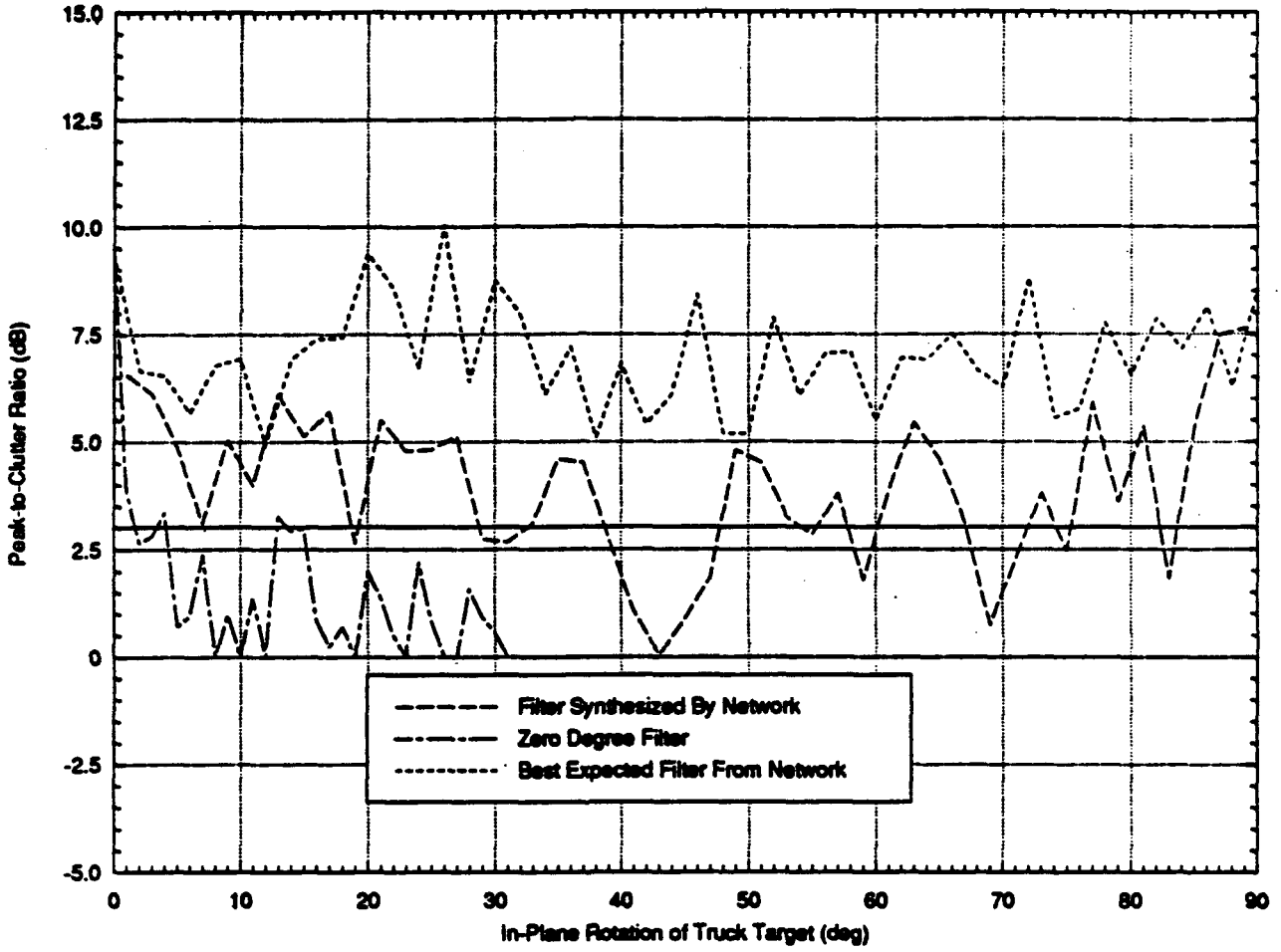


Figure 7.29 Filter synthesis results using a stretch and hammer neural network for the truck on the bushy background at truck rotation angles of 1, 3, ..., 89 degrees. The network was trained on 66 gray level, 142 gray level, and city70 backgrounds at truck rotation angles of 0, 2, ..., 90 degrees using as inputs 31 computed values from a target-centered 9 x 9 pixel grid.

7.6. Comparison of back-propagation and stretch and hammer neural network results

Both the stretch and hammer neural network and the back-propagation neural network perform a mapping form the input space to the output space. The stretch and hammer neural network uses a deformed least squares plane of dimensionality equal to the number of inputs to map the inputs to the outputs. The back-propagation neural network also uses a function of dimensionality equal to the number of inputs. However, this function is created by reducing the sum of the squared differences between the network outputs and the training outputs. This mapping function is deformed or altered by changing the values of the interconnection weights.

The results show that the stretch and hammer neural network did not perform as well as the back-propagation neural network using the same input/ output training set. However, the stretch and hammer neural network had a total of 138 + 31 + 1 = 170 adjustable parameters for each output coded filter value, whereas the back-propagation neural network had 0.5(31 x 200 + 200 + 1) = 3300.5 adjustable parameters for each such value. Thus the back-propagation neural network had an advantage of approximately 17 times in the number of adjustable parameters available for characterizing the rules for mapping inputs to outputs. However, the 3300.5 adjustable parameters were not independent because, as indicated in Section 6.4, the average number of adjustable parameters per output for the back-propagation neural network was 63,500/600 = 105.8. Current software limits the number of adjustable

parameters for the stretch and hammer neural network, but if this number could be increased to 3300 it is reasonable to expect that this network would have comparable or superior performance compared to the back-propagation neural network.

8. SUMMARY AND CONCLUSION

The feasibility of using neural networks to synthesize filters for a HAC system was demonstrated. Both stretch and hammer and back-propagation neural networks successfully synthesized filters. The stretch and hammer neural network trained more rapidly and had the advantage of exact learning. For the filter synthesis results obtained here the back-propagation neural network performed better than the stretch and hammer neural network, although the back-propagation neural network required approximately 40 hours to train on a 486-class 33 MHz desktop computer, whereas the stretch and hammer neural network required approximately 3 hours. However, training time and performance depended on the number of training examples and the number of hidden neurons. In a simple test where these variables were selected so that the number of independent adjustable parameters (or neural network weights) were the same, the stretch and hammer neural network both trained more rapidly and out-performed the back-propagation neural network.

A relatively new input scene binarization technique was used. This technique involved edge-enhancement prior to thresholding to retain more target information, thus providing a sufficient signal (target) to noise (background) ratio for pattern recognition for a variety of cluttered backgrounds.

TPAF filters were used for correlation. A filter was generated by binarizing a target using the same technique as for input scene binarization, performing a computer simulated Fourier transform on this binarized target (using TLA = 0 degrees to create a symmetric cosine BPOF), averaging 3 x 3 pixel grids, and multiplying the resulting 3 x 3 superpixel BPOF by a 10-60 pixel radius bandpass to create a TPAF with only 600 coded values. The neural networks used in this research were trained to produce these 600 values as outputs.

The training inputs to the neural networks were values determined from the input scene. An input scene sampling strategy was developed that provided a sufficient amount of representative information to the neural networks. A variety of sampling strategies were analyzed. The strategy that most successfully ignored background effects was a 32 wedge input scene sampling technique. Sampling the input scene using target-centered sampling grids implies previous knowledge of target position. This restriction limits the applicability of filter synthesis to target tracking or target confirmation tasks. Advances have been made in blob recognition which may enable neural network filter synthesis to be used for target recognition. If after binarization the center of a blob can be found (representing the center of the target in the input scene), then it should be possible to use an input plane sampling grid to neural network filter synthesis.

Pattern recognition by optical correlation relies on using the properties of the Fourier spectrum of a target. Unfortunately, the Fourier spectrum varies

when the target undergoes rotation, scale, aspect, or other distortions. A single filter can successfully correlate only a limited range of these distortions. Thus successful correlation requires a large number of filters to accommodate all possible orientations and scalings of a target. This research shows how neural network filter synthesis can accommodate in-plane rotation distortions. This filter synthesis approach may be feasible for accommodating other distortions if adequate input/ output training sets are used.

Using a larger filter bandpass should allow for improved neural network filter synthesis performance, because more neural network outputs could be employed. Correlation performance could also be improved by using every pixel to synthesize filters. Using every pixel also would require nine times more neural network outputs, but this increase could be accommodated by employing more than one network to produce the coded filter values. Correlation performance could also be improved by training a separate neural network to learn a complex binary amplitude pattern of the TPAF filter.

APPENDIX

Figure A.1 **Truck image.**

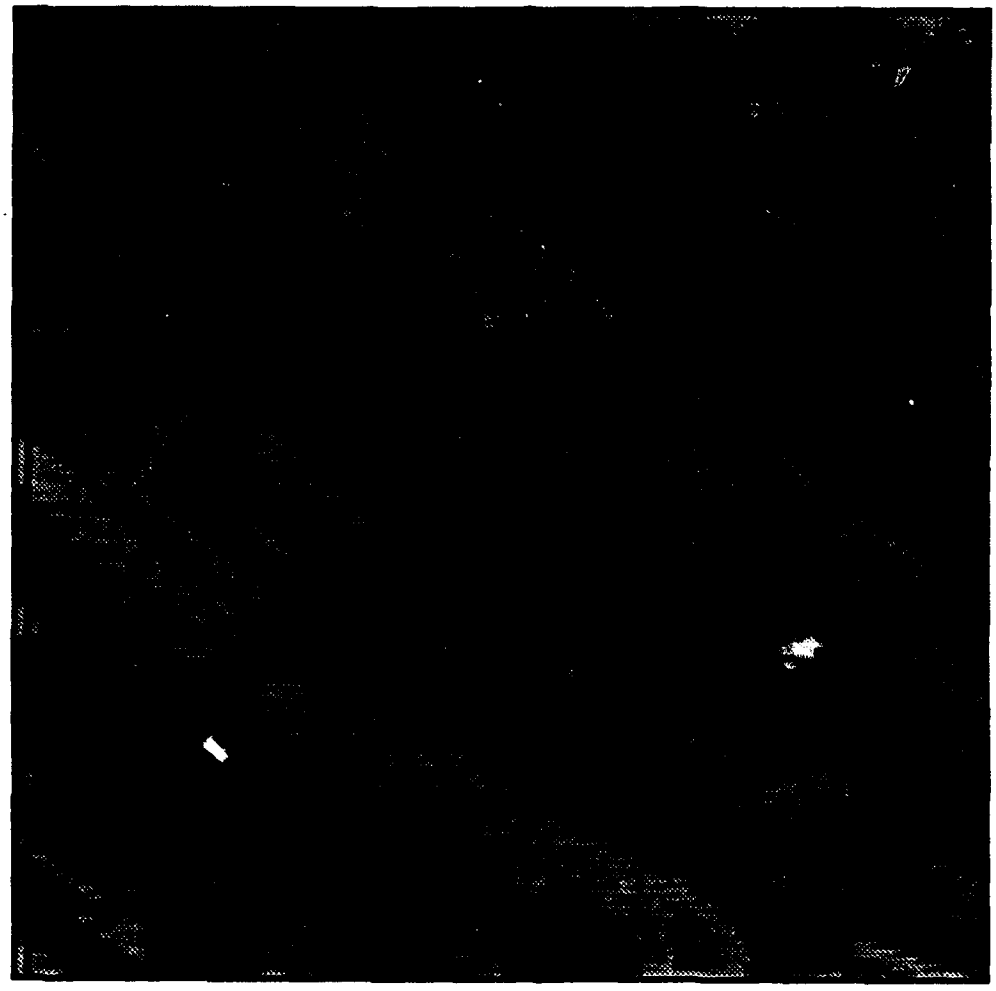


Figure A.2 City70 background.

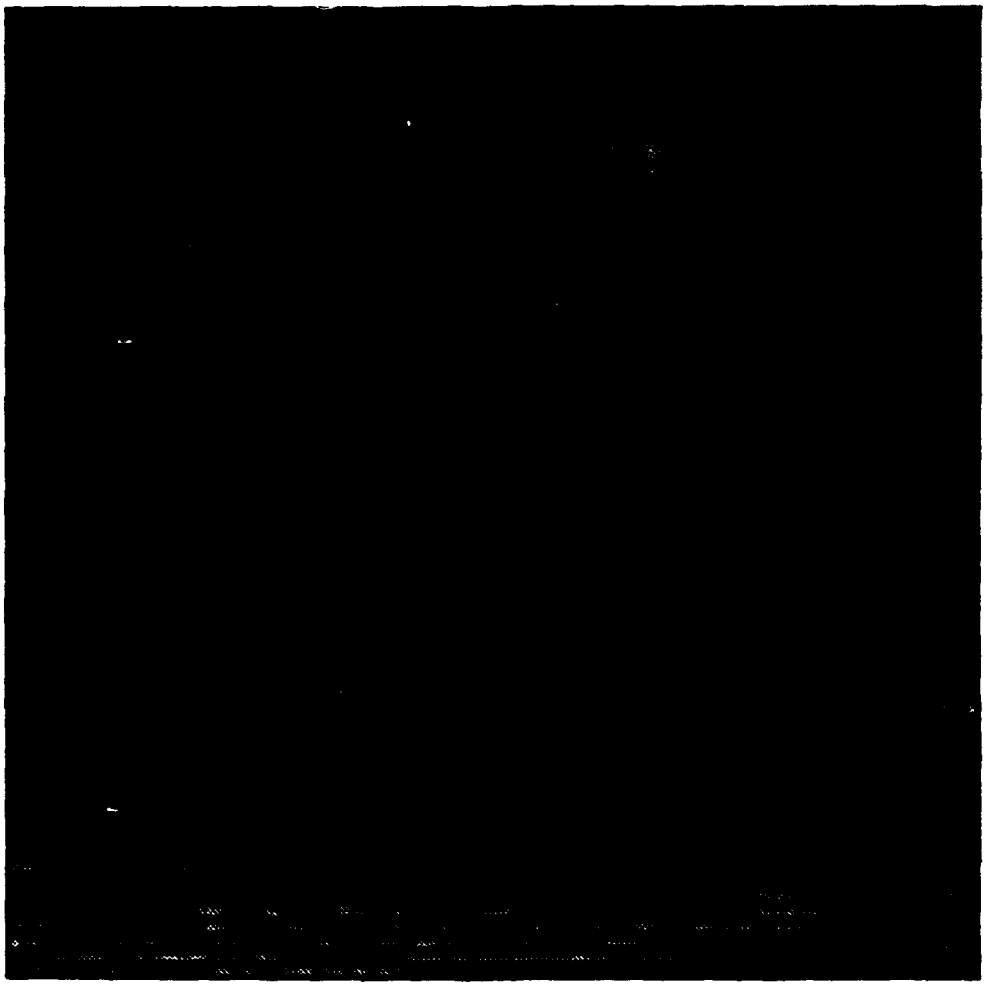


Figure A.3 Trbk70 background.

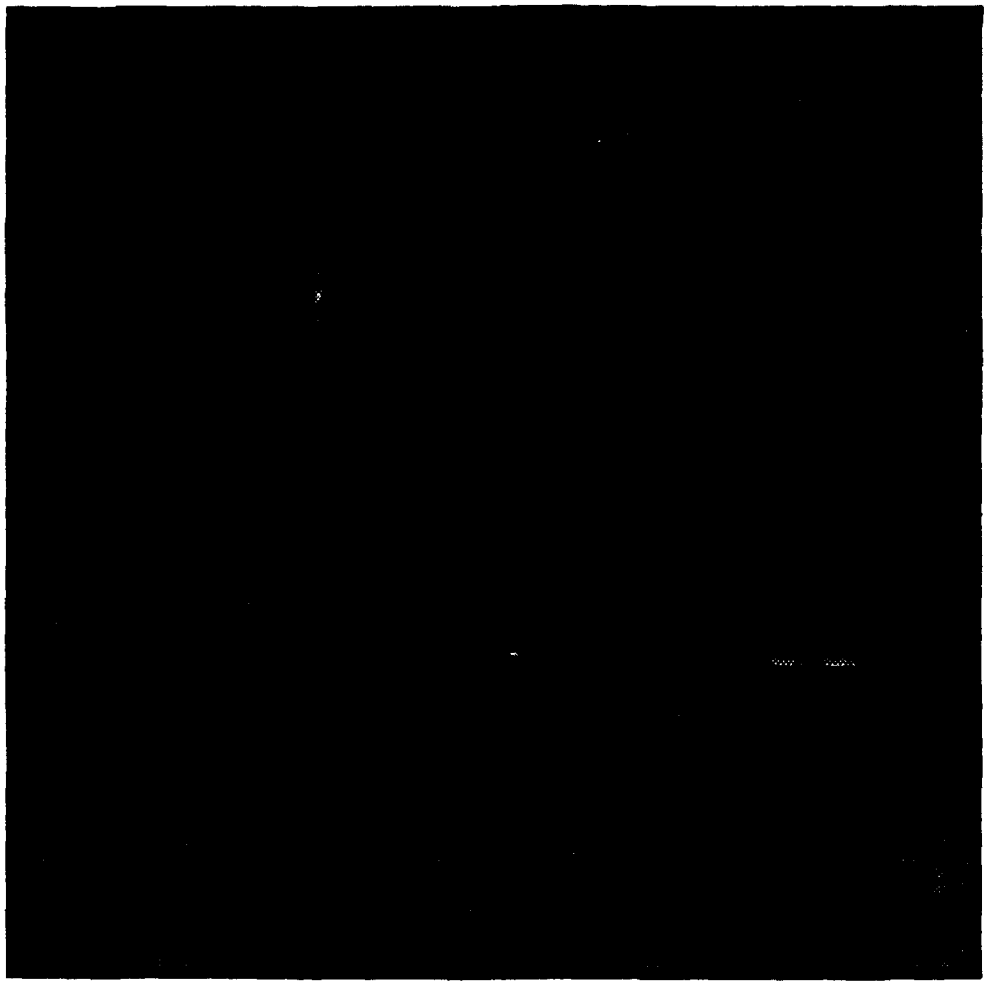


Figure A.4 Newbk80 background.

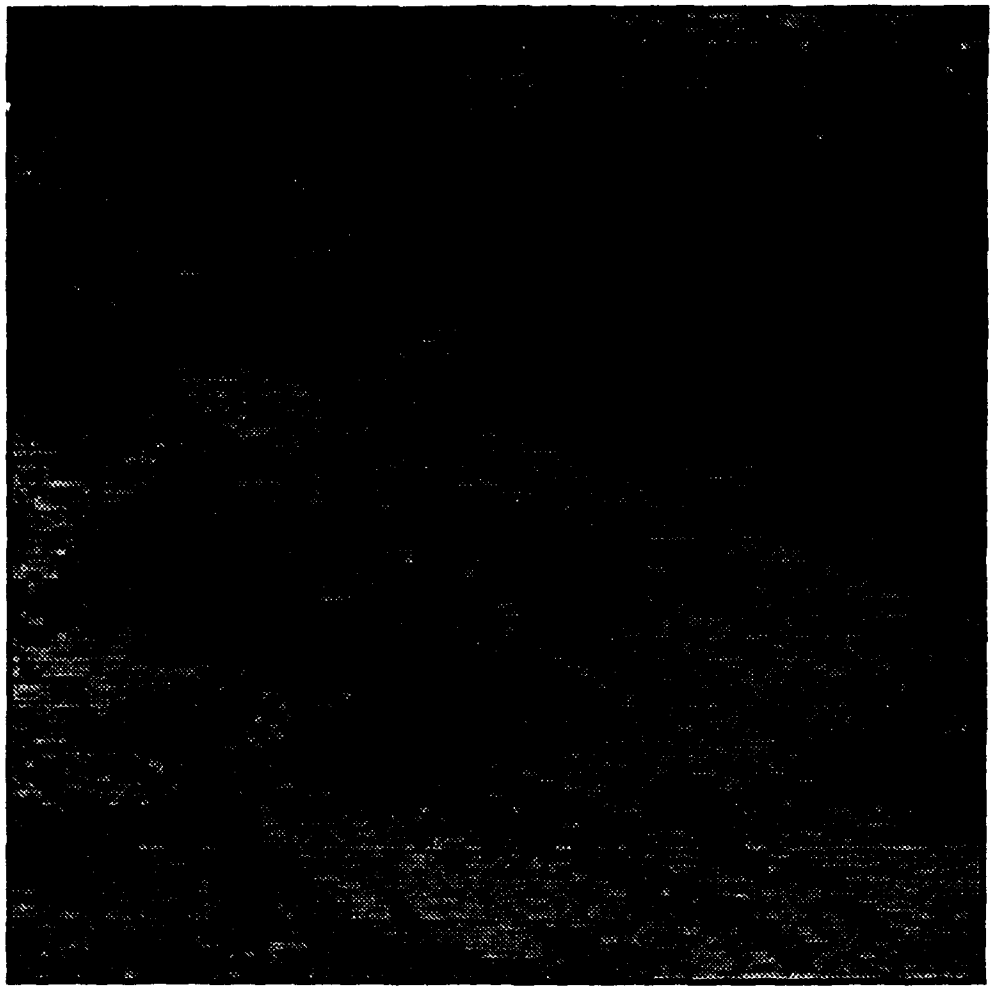


Figure A.5 Bushy background.

BIBLIOGRAPHY

1. A. B. VanderLugt, "Signal Detection by Complex Spatial Filtering," Radar Lab., Rept. No. 4594-22-T, University of Michigan, 1963.

2. J. W. Goodman, **Introduction to Fourier Optics**, McGraw Hill, 1968.

3. W. Ross, D. Psaltis, and R. Anderson, "Two Dimensional Magneto-optic Spatial Light Modulator for Signal Processing," Optical Engineering, Vol. 24, pp. 485-490, 1983.

4. B. Kast, M. Giles, S.Lindell, and D. Flannery, "Implementation of ternary phase-amplitude filters using a magneto-optic spatial light modulator," Applied Optics, Vol. 28, pp. 1044-46, 1989.

5. A. Oppenheim and J. Lim, "The Importance of Phase in Signals," Proc. IEEE, Vol. 69, pp. 529-541, 1981.

6. J. L. Horner and P. D. Gianino, "Phase-Only Matched Filtering," Applied Optics, Vol. 23, pp. 812-816, 1984.

7. D. L. Flannery and J. L. Horner, "Fourier Optical Signal Processors," Proc. IEEE, Vol. 77, pp. 1511-27, 1989.

8. D. Flannery, J. Loomis, and M. Milkovich, "Design Elements of Binary Phase-only Correlation Filters," Applied Optics, Vol. 27, pp. 4231-4235, 1988.

9. K. Fielding and J. Horner, "Clutter Effects on Optical Correlation," Proc. SPIE, Vol. 1151, pp. 130-137, 1990.

10. J. L. Horner and J. R. Legers, "Pattern Recognition with Binary Phase-only Filters," Applied Optics, Vol.24, pp. 609-611, 1985.

11. D. Psaltis, E. Paek, and S. Venkatesh, "Optical Image Correlation with a Binary Spatial Light Modulator," Optical Engineering, Vol. 23, pp. 698-704, 1984.

12. E. G. Olczak, "Neural Networks for the Hybrid Adaptive Correlator," Master's Thesis, University of Dayton, 1991.

13. G. W. Johnson, "Digital Image Processing and Neural Networks for Real-Time Target Recognition," Master's Thesis, University of Dayton, 1991.

14. D. E. Rumelhart and J. L. McClelland, "Parallel Distributed Processing: Exploration in the Microstructure of Cognition," MIT Press, 1986.

15. R. P. Lipmann, "An Introduction to Computing with Neural Nets," IEEE Acoustics Speech and Signal Processing, pp. 4-22, 1987.

COMPARISON OF RADIAL BASIS FUNCTION AND CARDINAL CUBIC SPLINE INTERPOLATION

Steven C. Gustafson, Troy A. Rhoadarmer, John S. Loomis, and Gordon R. Little
University of Dayton, 300 College Park, Dayton, Ohio 45469-0140

A key problem in the implementation of radial basis function neural networks (e.g., Moody and Darken, 1987) is the determination of basis function widths. A diagonal dominance technique has been described (Gustafson, et al. 1992a,b) and demonstrated (Gilbert and Gustafson, 1993) that determines these widths by relating them to neural network stability, i.e., the widths are selected so that small changes in the training data yield small changes in the neural network weights. Here this technique is investigated for one-variable (single-input) functions by comparing Gaussian radial basis function interpolation with cardinal cubic spline interpolation, which is the smoothest possible interpolation according to the least integrated squared second derivative criterion required by regularization theory (e.g., Poggio and Girosi, 1990).

A Gaussian radial basis function neural network interpolates m training points (x_i, y_i) using $f(x) = \Sigma_j c_j \exp[-(x - x_j)^2/\sigma_j^2]$, where the σ_j are basis function widths and the c_j are neural network weights. Once the σ_j are determined the c_j are found by solving m simultaneous linear equations in m unknowns $y_i = \Sigma_j c_j a_{ij}$, where $a_{ij} = \exp[-(x_i - x_j)^2/2\sigma_j^2]$. The diagonal dominance technique selects the σ_j such that the matrix $A = \{a_{ij}\}$ is diagonally dominant by an amount ε for each column, i.e., so that $\varepsilon = a_{jj} - \Sigma_{i \neq j} a_{ij}$ for all j. It is well known that the ∞-norm of A is $\|A\|_\infty = \max_i \Sigma_j |a_{ij}|$ and that the ∞-norm and the 2-norm are related by $\|A\|_2 \leq \sqrt{m}\ \|A\|_\infty$. Also, it has been shown that the ∞-norm of A^{-1} is $\|A^{-1}\|_\infty \leq 1/\varepsilon$ (Varah, 1975). Using $a_{ii} = 1$, the 2-norm condition number of A is thus $\kappa_2 = \|A\|_2 \|A^{-1}\|_2 \leq m(2 - \varepsilon)/\varepsilon$. Finally, stability defined by $1/r_c$ is bounded by $1/r_c \geq [(\kappa_2 r_y)^{-1} - 1]/2$ for $\kappa_2 r_y < 1$, where $r_c = [\Sigma_i (c_i' - c_i)^2/\Sigma_i c_i^2]^{1/2}$ is the fractional root-mean-square coefficient change, $r_y = [\Sigma_i (y_i' - y_i)^2/\Sigma_i y_i^2]^{1/2}$ is the fractional root-mean-square data output change, and the c_i change to c_i' if the y_i change to y_i' (e.g., Golub and Van Loan, 1989). Thus for positive ε the diagonal dominance technique ensures a bound on neural network stability. This technique (Gustafson, et al. 1992a) has been suggested for (Gustafson et al., 1992b) and successfully demonstrated in (Gilbert and Gustafson1993) image processing applications. For a specified ε determination of the σ_j using the diagonal dominance technique requires the solution of m independent nonlinear equations each one unknown, i.e., the solution of $\varepsilon = a_{jj} - \Sigma_{i \neq j} a_{ij}$ for σ_j is required for all j.

Figure 1a shows points between $x = -3$ and 3 for an impulse function at $x = 0$ that is randomly sampled at 31 points, where the maximum and minimum points satisfy $|x| > 7$. A $\varepsilon = 0$ Gaussian radial basis function curve and a cardinal cubic spline curve that interpolate these 31 points are also shown, where the spline curve is cardinal because it is independent of the (distant) end conditions. Figure 1b is a plot of the root-mean-square difference between these curves from $x = -3$ to 3 as a function of ε; it indicates that $\varepsilon = -0.19$ yields the optimum agreement (a plot of maximum absolute difference has nearly the same minimum). Figure 1c is a histogram of ε values that yield the optimum agreement for many sets of 31 random training points, where the x values are uniformly random and extend from $x < -7$ to $x > 7$ and where the y values are uniformly random from 0 to 1 between $x = -3$ and 3 and are constant from the minimum and maximum x within this range to $x < -7$ and $x > 7$, respectively.

These results indicate that Gaussian radial basis function neural networks should have basis function widths determined by diagonal dominance with a positive ε as close to zero as acceptable neural network stability permits. Figure 1c indicates that such neural networks are most likely to yield interpolation curves in optimal agreement with the maximum-smoothness interpolation curves determined by regularization or spline methods. However, unlike these methods, radial basis function techniques are readily applicable to neural networks with multiple inputs and nonuniform training points.

References

D. W. Gilbert and S. C. Gustafson, "Satellite Image Processing Using a Hammering Neural Network," *Proc. SPIE* Vol. 1970, No. 2, Orlando, FL, April 1993.

G. H. Golub and C. F. Van Loan, *Matri utations*, 2nd ed., Johns Hopkins Univ. Press, 1989.

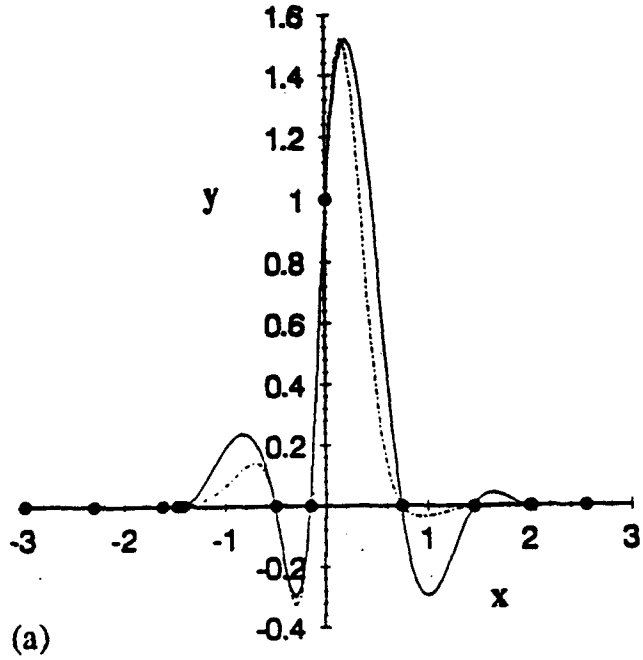
S. C. Gustafson, G. R. Little, M. A. Manzardo, and T. S. Puterbaugh, "Stretch and Hammer Neural Networks," *Proc. SPIE* Vol. 1710, pp. 43-52, Orlando, FL, April 1992a.

S. C. Gustafson, G. R. Little, J. S. Loomis, and T. S. Puterbaugh, "Optimal Reconstruction of Missing Pixel Images," *Appl. Optics*, Vol. 31, pp. 6829-6830, November 1992.

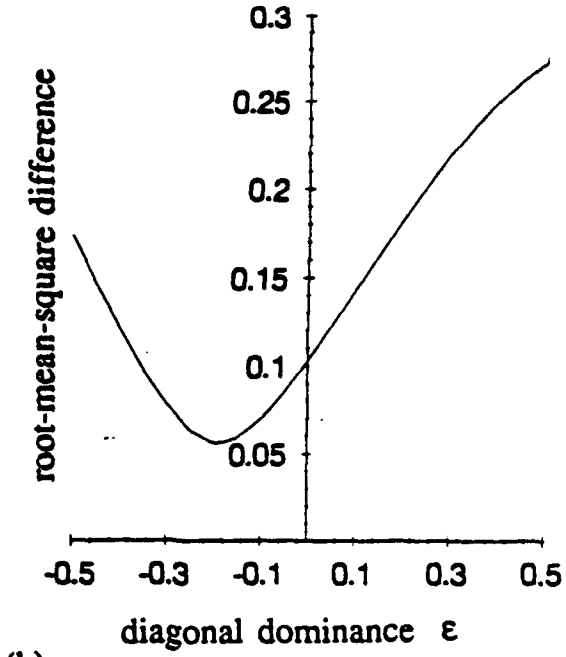
J. Moody and C. J. Darken, "Fast Learning in Networks of Locally-Tuned Processing Units," *Neural Computation*, Vol. 1, pp. 281-273, 1989.

T. Poggio and F. Girosi, "Networks for Approximation and Learning," *Proc. IEEE*, Vol. 78, pp. 1481-1497, September 1990.

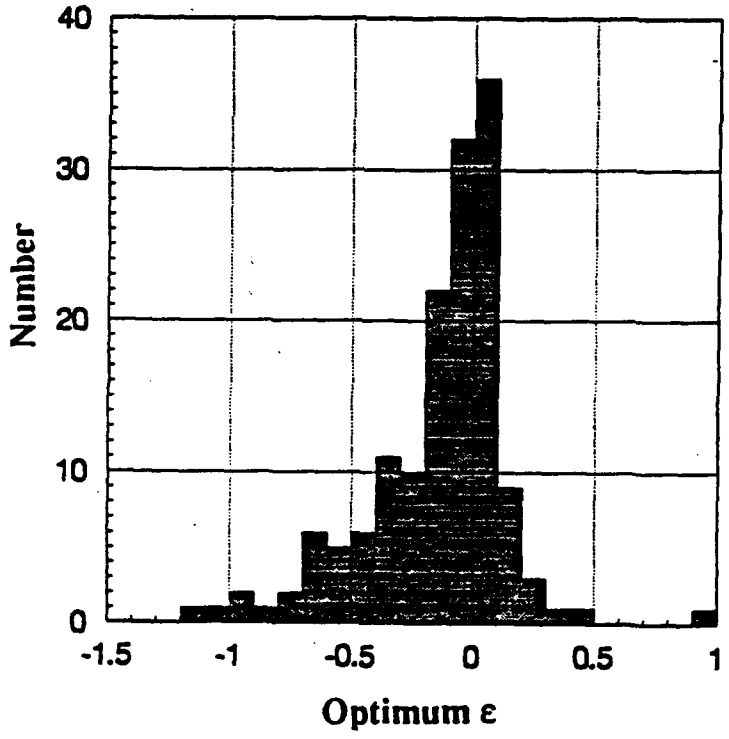
J. M. Varah, "A Lower bound for the Smallest Singular Value of a Matrix," *Linear Algebra and Its Applictions*, Vol. 11, pp. 3-5, January 1975.



(a)



(b)



(c)

Figure 1. (a). Randomly sampled impulse function with $\varepsilon = 0$ Gaussian radial basis function (broken) and cardinal cubic spline (solid) interpolation curves. (b). Plot of root-mean-square difference between these curves as a function of diagonal dominance ε. (c). Histogram of ε values that yield minimum root-mean-square differences between Gaussian radial basis function and cardinal cubic spline interpolation curves for random training points.

MISSION

OF

ROME LABORATORY

Rome Laboratory plans and executes an interdisciplinary program in research, development, test, and technology transition in support of Air Force Command, Control, Communications and Intelligence (C3I) activities for all Air Force platforms. It also executes selected acquisition programs in several areas of expertise. Technical and engineering support within areas of competence is provided to ESC Program Offices (POs) and other ESC elements to perform effective acquisition of C3I systems. In addition, Rome Laboratory's technology supports other AFMC Product Divisions, the Air Force user community, and other DOD and non-DOD agencies. Rome Laboratory maintains technical competence and research programs in areas including, but not limited to, communications, command and control, battle management, intelligence information processing, computational sciences and software producibility, wide area surveillance/sensors, signal processing, solid state sciences, photonics, electromagnetic technology, superconductivity, and electronic reliability/maintainability and testability.